BUUCTF笔记之Reverse(Java及android部分)

原创

KogRow 于 2021-06-03 19:16:24 发布 181 收藏 1
分类专栏： CTF 杂项 文章标签： reverse CTF

先把这最简单的拿下，将来或许会继续研究reverse

1.easyre

拖进winhex,直接搜flag：

```
00026DE0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00026DF0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00026E00  25 64 25 64 00 66 6C 61  67 7B 74 68 69 73 5F 49  %d%d flag{this_I
00026E10  73 5F 61 5F 45 61 53 79  52 65 7D 00 73 6F 72 72  s_a_EaSyRe} sorr
00026E20  79 2C 79 6F 75 20 63 61  6E 27 74 20 67 65 74 20  y,you can't get
```

2.Java逆向解密

下载下来一个字节码文件，拖进jdgui得到反编译的源码：

```
import java.util.ArrayList;
import java.util.Scanner;


public class Reverse
{
  public static void main(String[] args) {
    Scanner s = new Scanner(System.in);
    System.out.println("Please input the flag );
    String str = s.next();
    System.out.println("Your input is );
    System.out.println(str);
    char[] stringArr = str.toCharArray();
    Encrypt(stringArr);
  }

  public static void Encrypt(char[] arr) {
    ArrayList<Integer> Resultlist = new ArrayList<>();

    for (int i = 0; i < arr.length; i++) {
      int result = arr[i] + 64 ^ 0x20;
      Resultlist.add(Integer.valueOf(result));
    }
    int[] KEY = { 180, 136, 137, 147, 191, 137, 147, 191, 148, 136, 133, 191, 134, 140, 129, 135, 191, 65 }
    ArrayList<Integer> KEYList = new ArrayList<>();
    for (int j = 0; j < KEY.length; j++) {
      KEYList.add(Integer.valueOf(KEY[j]));
    }
    System.out.println("Result:");
    if (Resultlist.equals(KEYList)) {
      System.out.println("Congratulations);
    } else {
      System.err.println("Error);
    }
  }
}
```

看了一下是一道送分题，payload：

```
int[] KEY = { 180, 136, 137, 147, 191, 137, 147, 191, 148, 136, 133, 191, 134, 140, 129, 135, 191, 65 };
ArrayList<Integer> keys = new ArrayList<Integer>(KEY.length);
String s = "";
for(int i=0;i<KEY.length;i++) {
    s+=(char)(KEY[i]-64^0x20);
}
System.out.println(s);
```

得到flag:

<terminated> Main2 [Java Application] C:\Program Files\J

This_is_the_flag_!

<terminated> Main2 [Java Application] C:\Program Files\J

This_is_the_flag_!

3.[SWPU2019]Android1

现在的android题真是有毒啊。

直接上IDA里面F5之后的代码了：

```
int __cdecl Java_com_example_ndktest2_MainActivity_Encrypt(_JNIEnv *a1)
{
  char *v1; // eax
  char *v2; // eax
  char *v4; // [esp+24h] [ebp-B8h]
  char v5[13]; // [esp+2Dh] [ebp-AFh] BYREF
  char dest[6]; // [esp+3Ah] [ebp-A2h] BYREF
  char v7[128]; // [esp+40h] [ebp-9Ch] BYREF
  char v8[28]; // [esp+C0h] [ebp-1Ch] BYREF

  strcpy(dest, "flag{");
  strcpy(&v5[7], "wllm");
  strcpy(v8, "welcome");
  strcpy(v5, "newbee");
  base64_encode(v5, v7);
  v4 = strcat(dest, &v5[7]);
  v1 = strcat(v8, v5);
  v2 = strcat(v4, v1);
  return _JNIEnv::NewStringUTF(a1, v2);
}
```

分析一波：先把"flag{"赋值给dest

v5有13个字节，在strcpy(&v5[7], "wllm");结束后v5在内存里是这样的：

_ _ _ _ _ _ _ w l l m _ _

然后把"welcome"赋值给v8

然后把"newbee"赋值给v5

所以这时v5是这样的

n e w b e e _ w l l m _ _

然后对v5进行一次base64操作得到：v7="bmV3YmVld2xsbQ=="

接下来就是拼接：

v4="flag{wllm"

v1="welcomenewbeewllm"

v2="flag{wllmwelcomenewbeewllm"

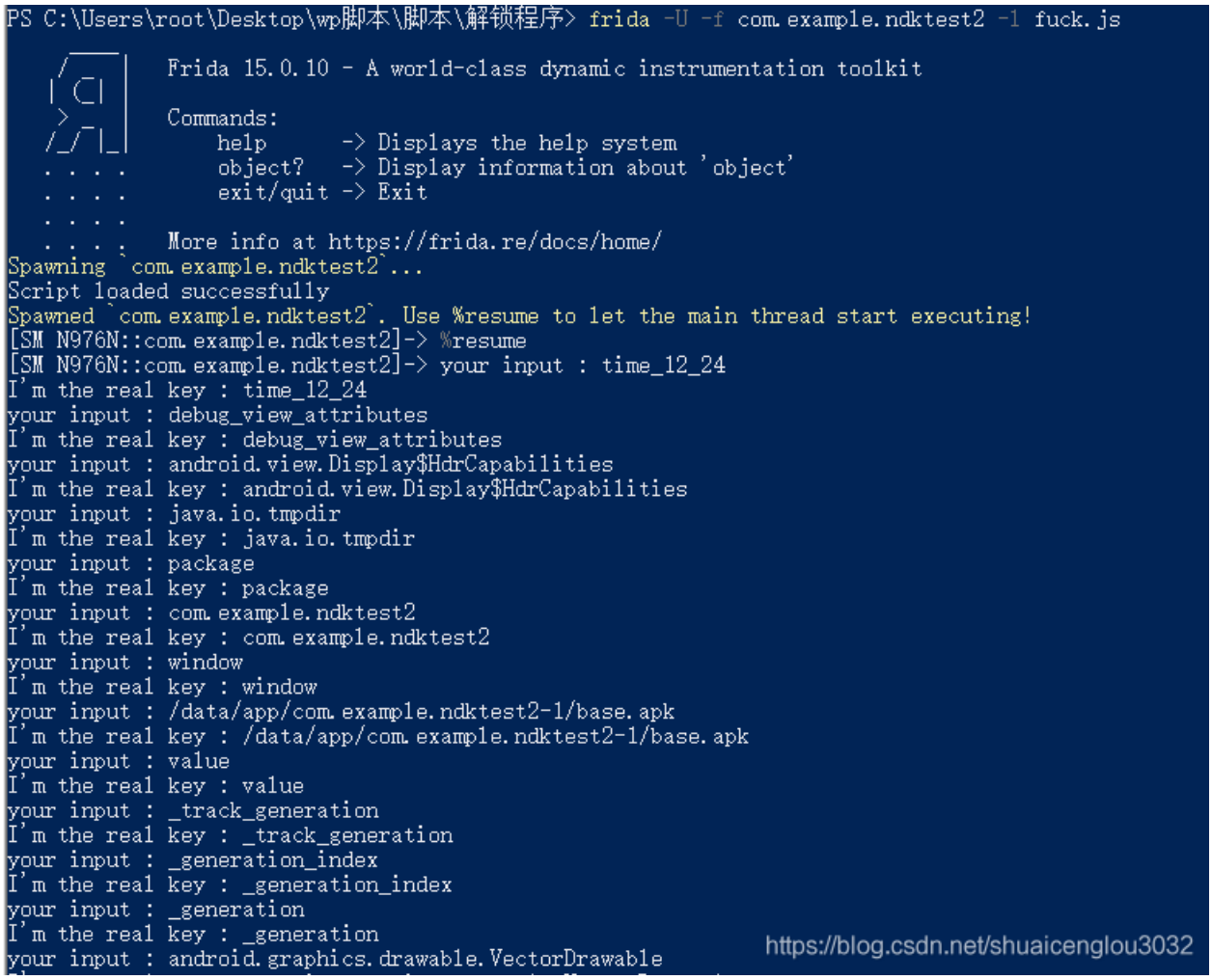
然后失败了。。。。。

试一下frida hook函数：

写一个js脚本劫持java的equals函数：

```
console.log("Script loaded successfully ");
Java.perform(function x() {//hook思路3：系统打桩，监控系统api equals方法的调用，每调用一次equals方法就打印出一次equ
    var String = Java.use('java.lang.String')//定位到要hook的类名
    String.equals.implementation = function (arg1) {//equals就是我们要hook的函数
    console.log("your input : " + this.toString());
    console.log("I'm the real key : " + arg1);//打印出来真实的解锁码
    var ret = this.equals(arg1);
    return ret;//返回值
    }
});
```

然后在攻击机上执行命令frida -U -f com.example.ndktest2 -l fuck.js：



然后输入%resume让程序开始执行。

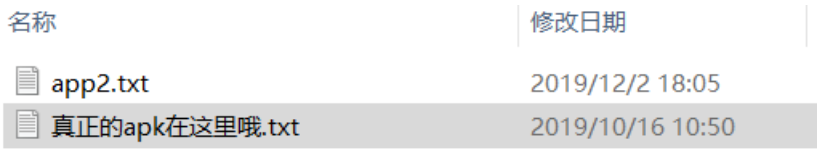
然后在验证框里面随便输入，equals方法被调用时就被劫持了，打印出来NDK返回的真实flag:

flag{YouaretheB3ST}。

4.[SWPU2019]Android2

这题也是神坑，到现在都没做出来，先把进展记录一下。

这题下载下来的APK并不是真的，真正的APK在假的APK的res文件夹里面。



把APK改成ZIP之后解压得到

把app2.txt改成apk之后得到真正的APK。拖进jadx分析一波MainActivity：

```
package com.example.thousandyearsago;

import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    /* renamed from: a */
    final long f32a = 1000000000;
    private Handler handler;

    public native String StringFromJNI();

    public native String stringFromJNI();

    static {
        System.loadLibrary("native-lib");
    }

    /* access modifiers changed from: protected */
    @Override // androidx.core.app.ComponentActivity, androidx.appcompat.app.AppCompatActivity, androidx.fr
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(C0272R.layout.activity_main);
        this.handler = new Handler();
        ((Button) findViewById(C0272R.C0274id.button)).setOnClickListener(this);
        this.handler.postDelayed(new Inner(), 1000000000);
    }

    public void onClick(View v) {
        if (v.getId() == C0272R.C0274id.button) {
            Toast.makeText(this, "好像卡死了啊！！！", 0).show();
        }
    }

    class Inner implements Runnable {
        Inner() {
        }

        public void run() {
            if (1000000000 <= 1000000) {
                MainActivity mainActivity = MainActivity.this;
                Toast.makeText(mainActivity, mainActivity.StringFromJNI(), 1).show();
            }
        }
    }
}
```

不出意外又是native。这题的MainActivity在启动之后监听按钮，点击则弹出“好像卡死了啊”，同时延时277个小时（100万秒）启动一个线程，这个线程只做一件事，如果10亿小于100万则得到flag。这是一个假条件，现在需要想办法把延时277小时启动给改掉，同时让线程里的代码能够执行。

5.[SCTF2019]Strange apk

这题下载下来先看Manifest.xml配置文件找程序入口：

```
25 ity>
32 ty android:name="sctf.demo.myapplication.s">
33 tent-filter>
34  <action android:name="sctf.demo.myapplication.M
36  <category android:name="android.intent.category.
37  <category android:name="sctf.demo.myapplication.
33 ntent-filter>
```


demo包下，但是反编译出来的包里并没有demo包，只有hello：

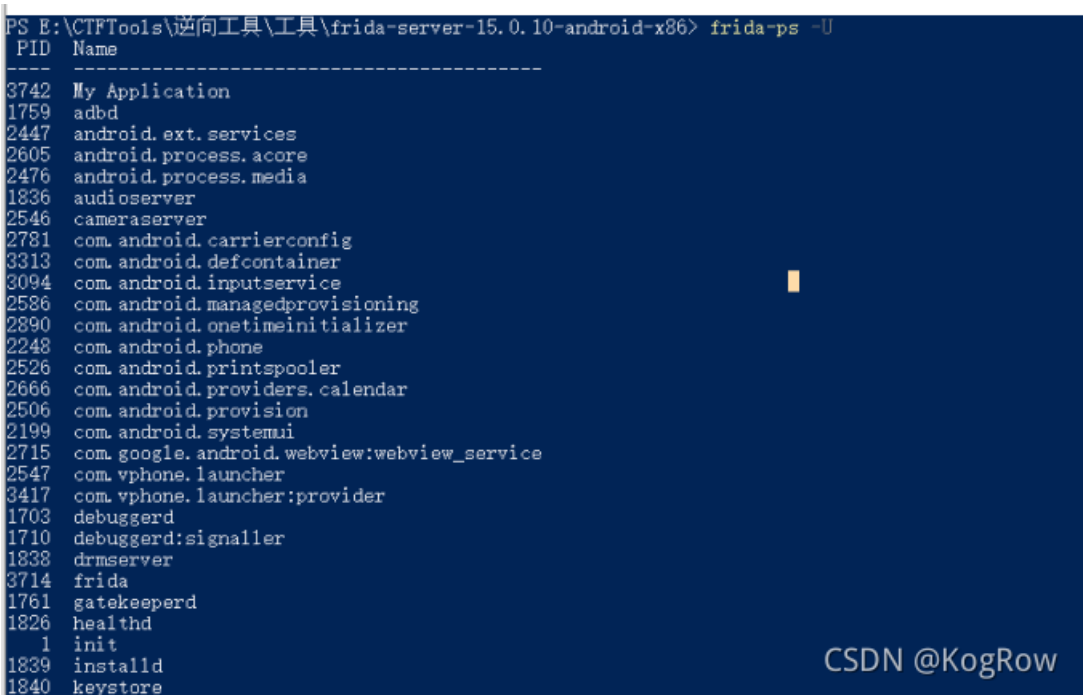
这里就又有一个新知识点了：考虑APP动态释放文件的可能性（参考PC下的压缩壳）。

因此考虑使用反射大师把运行时的dex文件dump下来分析。

其他writeup都是说安装Xposed和反射大师把实际运行的dex文件dump下来分析，但是我实际操作下来发现windows操作系统下，只用模拟器的情况下根本没有任何操作性。

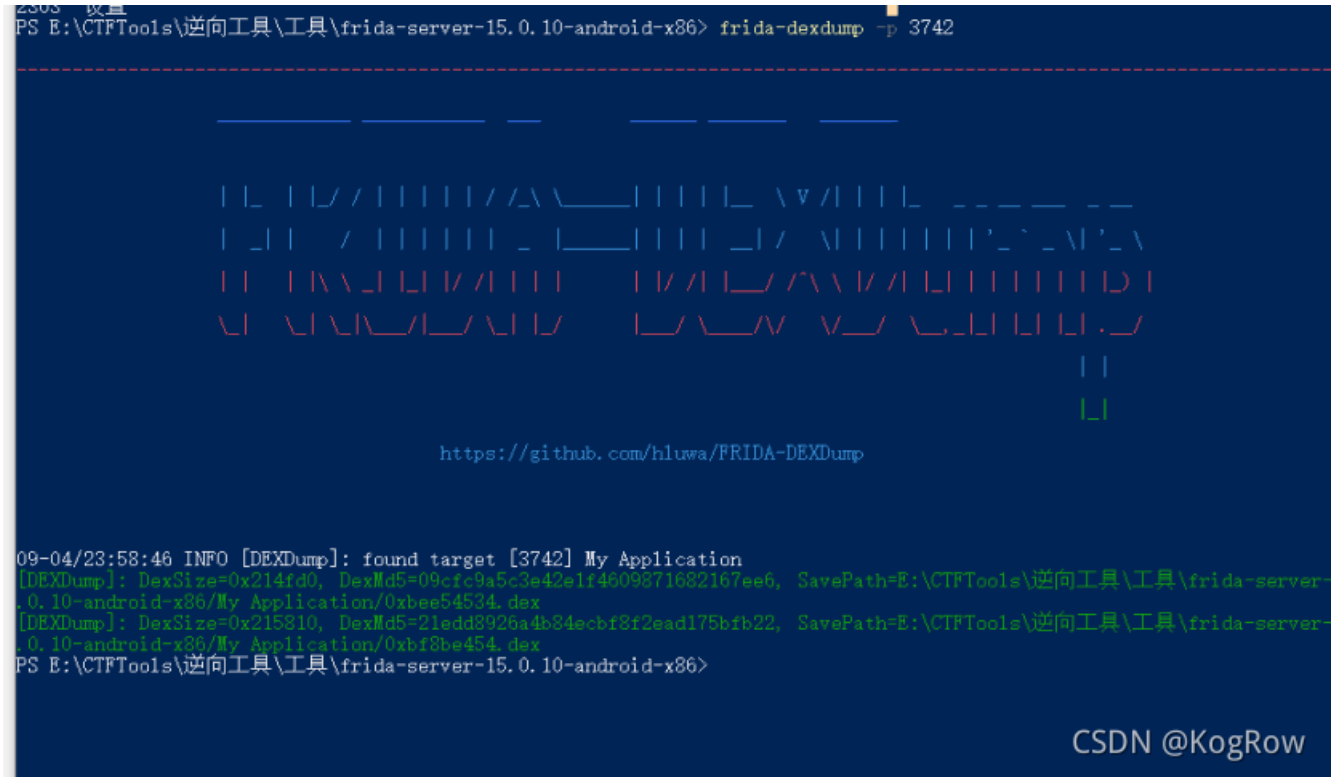
为了安装反射大师，我尝试了逍遥、雷电、mumu、夜神的各个版本的安卓，全是坑，根本用不成。我装这些模拟器之后，处理器型号统统是x86，在系统版本为7.1时，官方安装器和神盾装的Xposed框架无一例外会导致模拟器重启卡死在99%然后崩溃。在系统版本为5.1时，反射大师也正常装进去了，但是在导出dex文件的时候提示没有写权限，推测是安卓5.0和7.0挂载磁盘的方式不一样导致的，这个根本没法解决。

搞了半天还是回归神器frida吧：



frida-ps -U查看目标进程的PID为3742，所以下一步直接frida-dexdump -p 3742

把dex文件dump下来，这不比什么反射大师香吗？！！！：


PS E:\CTFTools\逆向工具\工具\frida-server-15.0.10-android-x86> frida-dexdump -p 3742
https://github.com/hluwa/FRIDA-DEXDump
09-04/23:58:46 INFO [DEXDump]: found target [3742] My Application
[DEXDump]: DexSize=0x214fd0, DexMd5=09cfc9a5c3e42e1f4609871682167ee6, SavePath=E:\CTFTools\逆向工具\工具\frida-server-15.0.10-android-x86/My Application/0xbee54534.dex
[DEXDump]: DexSize=0x215810, DexMd5=21edd8926a4b84ecbf8f2ead175bfb22, SavePath=E:\CTFTools\逆向工具\工具\frida-server-15.0.10-android-x86/My Application/0xbf8be454.dex
PS E:\CTFTools\逆向工具\工具\frida-server-15.0.10-android-x86>
CSDN @KogRow

拖进jadx分析：

```
package sctf.demo.myapplication;

import android.content.Intent;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class s extends AppCompatActivity {
    /* access modifiers changed from: protected */
    @Override // android.support.v7.app.AppCompatActivity, android.support.v4.app.SupportActivity, android.
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        findViewById(R.id.textView);
        final EditText ed = (EditText) findViewById(R.id.editText);
        ((Button) findViewById(R.id.button)).setOnClickListener(new View.OnClickListener() {
            /* class sctf.demo.myapplication.s.AnonymousClass1 */

            public void onClick(View v) {
                String s1 = BuildConfig.FLAVOR;
                String s2 = BuildConfig.FLAVOR;
                int i = 0;
                String s = ed.getText().toString();
                if (s.length() == 30) {
                    while (i < 12) {
                        s1 = s1 + s.charAt(i);
                        i++;
                    }
                    String s12 = f.sctf(s1);
                    while (i < 30) {
                        s2 = s2 + s.charAt(i);
                        i++;
                    }
                    if (s12.equals("c2N0ZntXM2xjMG1l")) {
                        Intent intent = new Intent();
                        intent.putExtra("data_return", s2);
                        s.this.setResult(-1, intent);
                        s.this.finish();
                        return;
                    }
                    Toast.makeText(s.this.getApplicationContext(), "something wrong", 1).show();
                    return;
                }
                Toast.makeText(s.this.getApplicationContext(), "something wrong", 1).show();
            }
        });
    }
}
```

这段代码也不难，flag长度为30，前12位进行base64生成，值为c2N0ZntXM2xjMG1l，解码得sctf{W3lc0me，后18位使用 Intent.putExtra()方法传了出去：

intent是Android程序中各组件之间进行交互的一种重要方式，一般被用来启动活动、启动服务以及发送广播等；intent在启动Activity这时候就需要用到putExtra()方法。intent中提供一系列的putExtra()方法的重载，可以把想要传递的数据暂存在intent中，当另一个putExtra("A", B)方法中，AB为键值对，第一个参数为键名，第二个参数为键对应的值，这个值才是真正要传递的数据。

搜索键“data_return”得到s2被传到了这里：

```
38              MessageDigest md = MessageDigest.getInstance("MD5");
39              md.update("syclover".getBytes());
40              key = new BigInteger(1, md.digest()).toString(16);
            } catch (Exception e) {
42              e.printStackTrace();
            }
47          if (f.encode(data.getStringExtra("data_return"), key).equals("~8t808_8A8n848r808i8d8-8w808r8l8d8}8")) {
48              tv.setVisibility(0);
49              bu.setVisibility(4);
59              return;
            }
```


看一下encode方法：

```
public static String encode(String str, String key) {
        int s = str.length();
        int c = key.length();
        StringBuilder t = new StringBuilder();
        for (int f = 0; f < s; f++) {
            t.append(str.charAt(f));
            t.append(key.charAt(f / c));
        }
        return t.toString();
    }
```

这里的key是这么来的：

```
MessageDigest md = MessageDigest.getInstance("MD5");
md.update("syclover".getBytes());
key = new BigInteger(1, md.digest()).toString(16);
```

自己运行一下得到key=8bfc8af07bca146c937f283b8ec768d4:

```
 3 import java.io.ByteArrayOutputStream;
11
12 public class Main {
13     public static void main(String[] args) throws NoSuchAlgorithmException {
14         MessageDigest md = MessageDigest.getInstance("MD5");
15         md.update("syclover".getBytes());
16         String key = new BigInteger(1, md.digest()).toString(16);
17         System.out.println(key);
18     }
19
20 }
21
```

Markers Properties Servers Data Source Explorer Snippets Console Progress Search

<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (2021年9月5日 上午12:25:03)

8bfc8af07bca146c937f283b8ec768d4

encode方法就是对stf的每一位进行遍历，然后在中间穿插8:

```
    public static void main(String[] args) throws NoSuchAlgorithmException {
        MessageDigest md = MessageDigest.getInstance("MD5");
        md.update("syclover".getBytes());
        String key = new BigInteger(1, md.digest()).toString(16);
        int s = 18;
        int c = key.length();
        StringBuilder t = new StringBuilder();
        for (int f = 0; f < s; f++) {
            System.out.println(key.charAt(f / c));
        }
        System.out.println(key);
    }

}
```

Markers Properties Servers Data Source Explorer Snippets Console Progress Search

<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (2021年9月5日 上午12:28:37)

```
8
8
8
8
8
8
8
8
8bfc8af07bca146c937f283b8ec768d4
```

截图(Alt + A)

所以把~8t808_8A8n848r808i8d8-8w808r8l8d8}8里面偶数位的8去掉就行：

~t0_An4r0id-w0rld}

得到佛莱格：flag{W3lc0me~t0_An4r0id-w0rld}

这道题的启示是，不能迷信查壳工具，我开始先用查壳工具没查出壳来，以为没有壳，被耍得团团转。实际上进来先找配置文件看程序入口，如果入口的类和反编译出来的不一样，那十有八九就是有壳，直接frida-dexdump把运行中释放出来的dex文件dump下来分析。

6.简单注册器

把apk拖进jadx，flag直接就在java层直接给出：

```
int flag = 1;
        if (flag == 1) {
            char[] x = "dd2940c04462b4dd7c450528835cca15".toCharArray();
            x[2] = (char) ((x[2] + x[3]) - 50);
            x[4] = (char) ((x[2] + x[5]) - 48);
            x[30] = (char) ((x[31] + x[9]) - 48);
            x[14] = (char) ((x[27] + x[28]) - 97);
            for (int i = 0; i < 16; i++) {
                char a = x[31 - i];
                x[31 - i] = x[i];
                x[i] = a;
            }
            System.out.println(String.valueOf(x));
```

运行拿flag

7.findit

具体同上。代码：

```
final char[] a = {'T', 'h', 'i', 's', 'I', 's', 'T', 'h', 'e', 'F', 'l', 'a', 'g', 'H', 'o', 'm', 'e'};
        final char[] b = {'p', 'v', 'k', 'q', '{', 'm', '1', '6', '4', '6', '7', '5', '2', '6', '2', '0', '
  char[] x = new char[17];
        char[] y = new char[38];
        for (int i = 0; i < 17; i++) {
            if ((a[i] < 'I' && a[i] >= 'A') || (a[i] < 'i' && a[i] >= 'a')) {
                x[i] = (char) (a[i] + 18);
            } else if ((a[i] < 'A' || a[i] > 'Z') && (a[i] < 'a' || a[i] > 'z')) {
                x[i] = a[i];
            } else {
                x[i] = (char) (a[i] - '\b');
            }
        }
            for (int i2 = 0; i2 < 38; i2++) {
                if ((b[i2] < 'A' || b[i2] > 'Z') && (b[i2] < 'a' || b[i2] > 'z')) {
                    y[i2] = b[i2];
                } else {
                    y[i2] = (char) (b[i2] + 16);
                    if ((y[i2] > 'Z' && y[i2] < 'a') || y[i2] >= 'z') {
                        y[i2] = (char) (y[i2] - 26);
                    }
                }
            }
            System.out.println(String.valueOf(y));
```

运行得flag

8.rsa

openssl解析公钥

```
tom@kali:~/视频$ openssl rsa -pubin -text -modulus -in pub.key
RSA Public-Key: (256 bit)
Modulus:
    00:c0:33:2c:5c:64:ae:47:18:2f:6c:1c:87:6d:42:
    33:69:10:54:5a:58:f7:ee:fe:fc:0b:ca:af:5a:f3:
    41:cc:dd
Exponent: 65537 (0x10001)
Modulus=C0332C5C64AE47182F6C1C876D42336910545A58F7EEFEFC0BCAAF5AF341CCDD
writing RSA key
-----BEGIN PUBLIC KEY-----
MDwwDQYJKoZIhvcNAQEBBQADKwAwKAIhAMAzLFxkrkcYL2wch21CM2kQVFpY9+7+
/AvKr1rzQczdAgMBAAE=
-----END PUBLIC KEY-----
```


然后yafu分解大数。

然后生成私钥文件：

```
tom@kali:~/视频$ python3 a.py -f PEM -o key.key -p 285960468890451637935629440372639283459 -q 3
040087416046019244943281559752724184 63 -e 65537
Using (p, q) to initialise RSA instance

n =
c0332c5c64ae47182f6c1c876d42336910545a58f7eefefc0bcaaf5af341ccdd

e = 65537 (0x10001)

d =
b378155840fb2b8fbdd869db5b7e91994f1ece256ee1175ec2c2bd3a4a795ad1

p = 285960468890451637935629440372639283459 (0xd721fba58aa2cccf65862d209fd03903)

q = 304008741604601924494328155975272418463 (0xe4b5f4318e91babbe8d7a8e91680cc9f)

Saving PEM as key.key
tom@kali:~/视频$ openssl rsautl -decrypt -inkey key.key -in flag.enc -out flag
```


用私钥文件解密，得到flag:

ag{decrypt_256}