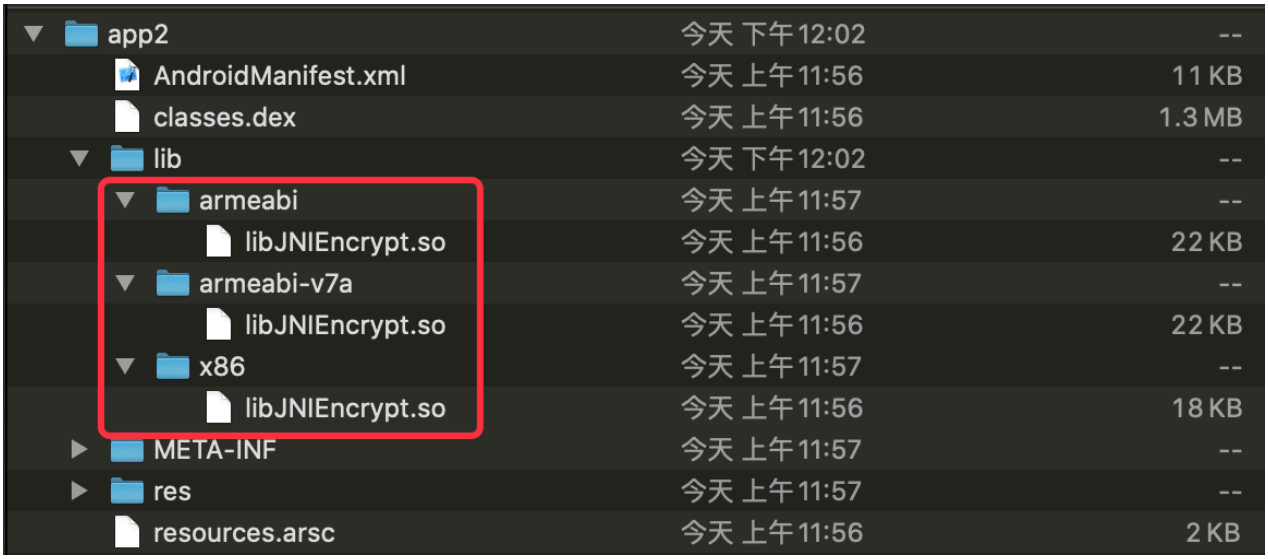
app2

> 附件地址： https://adworld.xctf.org.cn/media/task/attachments/30e5fb52c9134abbb4b80efc44ee4ec3.apk

（自己解题的时候比较懒，一般懒得去将apk安装。采取直接逆的方式。上一篇可能写的不是太详细，这篇再详细说说）

初探-分析

首先拿到一道安卓题目，我会解压apk，看一下lib文件夹中，是否有东西。一般来说，如果没有的话，题目就是较为简单的，分析java代码即可，如果有，还需要我们去逆向so文件。

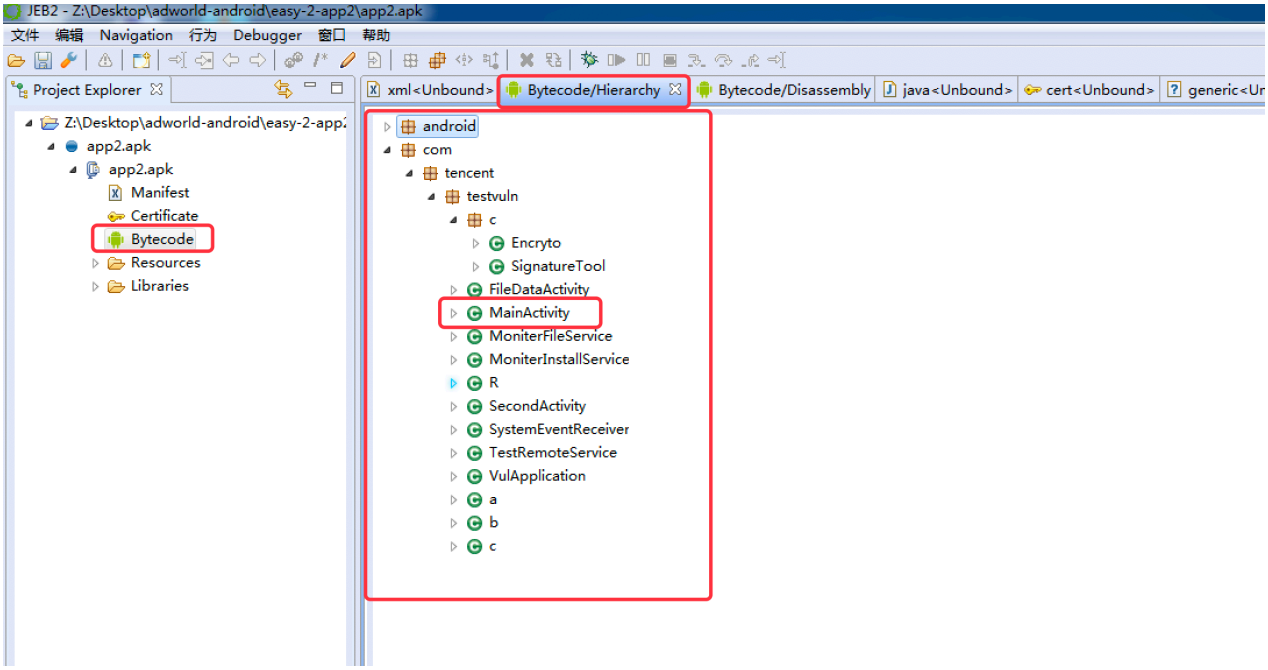


可以看到，在这个题目中，确实是存在so文件的。（大家不要在意那三个文件夹，他表示在不同架构下，去执行的so文件。

此时可以猜测这道题目的做法，首先分析java代码，找到关键逻辑，进而继续分析so。

开始解题-java

将apk文件，拖入JEB中。点击Bytecode，可以看到我们这个程序的结构。



此时我们需要关注MainActivity，这个一般是我们的程序入口。 点击进去，然后右键Decompile进行反编译。

分析代码，可以看到onClick这里，获取了c,d空间的文本内容。猜测是获取我们输入的两个字符串。

```
    public void onClick(View arg6) {
        switch(arg6.getId()) {
            case 2131165187: {
                if(this.c.getText().length() != 0 && this.d.getText().length() != 0) {
                    String v0 = this.c.getText().toString();
                    String v1 = this.d.getText().toString();
                    Log.e("test", v0 + " test2 = " + v1);
                    Intent v2 = new Intent(((Context)this), SecondActivity.class);
                    v2.putExtra("ili", v0);
                    v2.putExtra("lil", v1);
                    this.startActivity(v2);
                    return;
                }

                Toast.makeText(((Context)this), "不能为空", 1).show();
                break;
            }
        }
    }
```

进而保存为ili，lil传入给了SecondActivity。

然后我们继续跟进SecondActivity

```
protected void onCreate(Bundle arg6) {
    super.onCreate(arg6);
    this.setContentView(2130903041);
    Intent v0 = this.getIntent();
    String v1 = v0.getStringExtra("ili");
    String v2 = v0.getStringExtra("lil");
    if(Encryto.doRawData(this, v1 + v2).equals("VEIzd/V2UPYNdn/bxH3Xig==")) {
        v0.setAction("android.test.action.MoniterInstallService");
        v0.setClass(((Context)this), MoniterInstallService.class);
        v0.putExtra("company", "tencent");
        v0.putExtra("name", "hacker");
        v0.putExtra("age", 18);
        this.startActivity(v0);
        this.startService(v0);
    }

    SharedPreferences$Editor v0_1 = this.getSharedPreferences("test", 0).edit();
    v0_1.putString("ilil", v1);
    v0_1.putString("lili", v2);
    v0_1.commit();
}
```

可以发现在这里将我们传入的字符串取出，并进行加密，然后与一个加密字符串做比对操作。

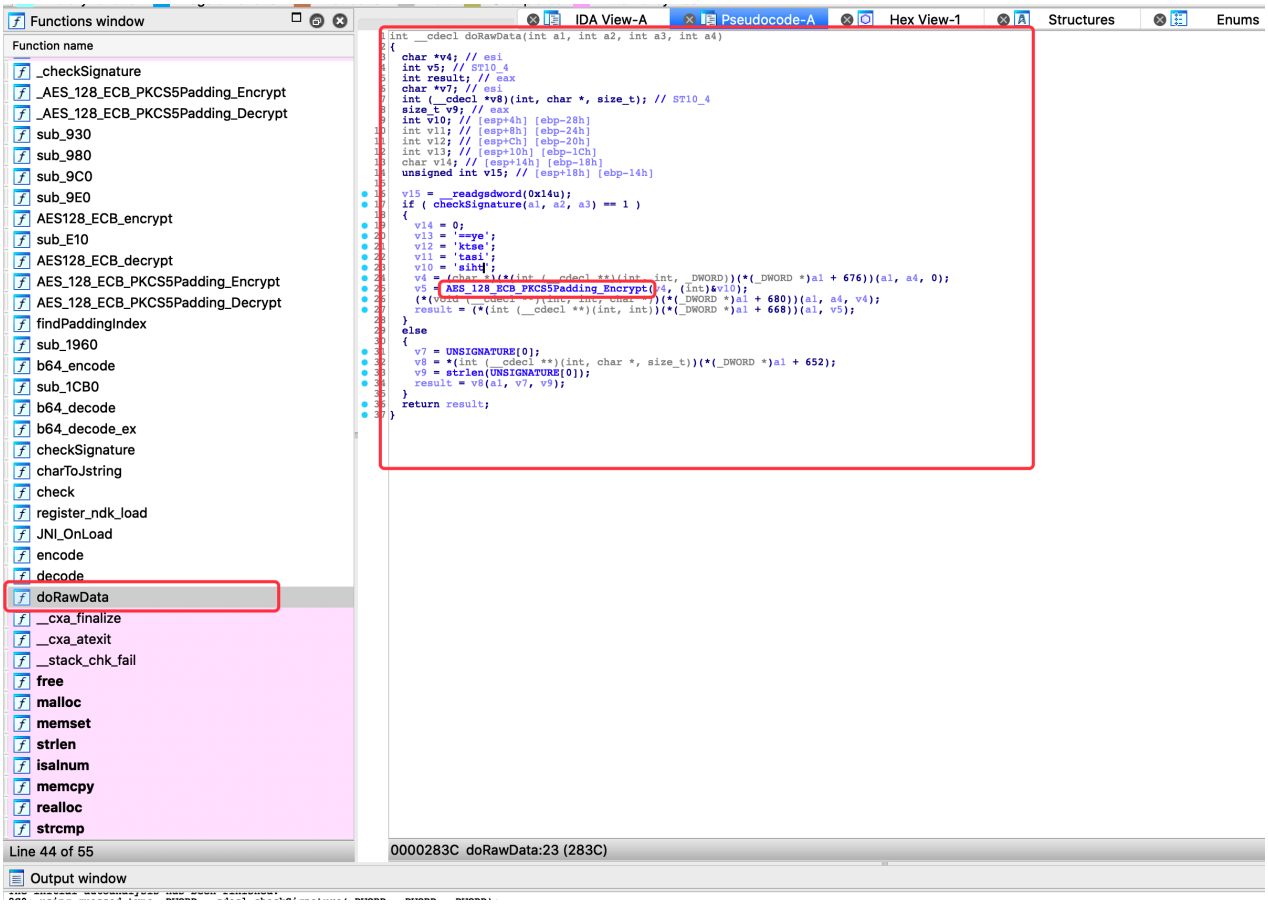
此时我们需要继续跟一下加密，看一下这里是如何实现的，以及我们该怎么逆向其代码。

```
package com.tencent.testvuln.c;

public class Encryto {
    static {
        System.loadLibrary("JNIEncrypt");
    }

    public Encryto() {
        super();
    }

    public native String HelloLoad() {
    }

    public static native int checkSignature(Object arg0) {
    }

    public static native String decode(Object arg0, String arg1) {
    }

    public static native String doRawData(Object arg0, String arg1) {
    }

    public static native String encode(Object arg0, String arg1) {
    }
}
```

可以看到，到这里，正式进入到了我们的so文件中。

开始解题-so

将三个so文件中的任意一个拖入ida中，



然后根据我们刚刚已知的doRawData关键字，找到相应的函数。

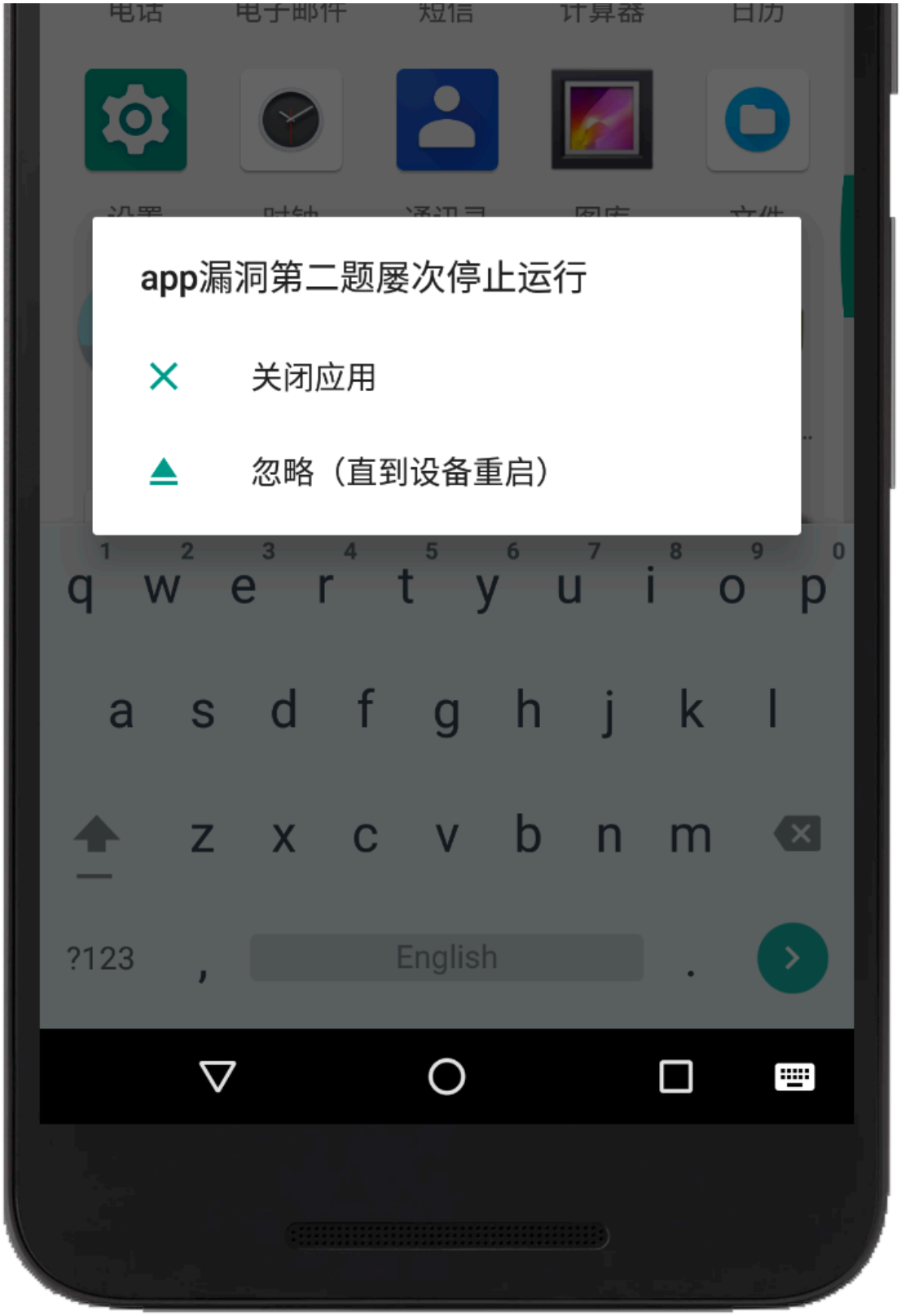
分析，可以发现，应该是对我们传入的数据做了AES加密。

v4应该是我们传入的值，v10则是秘钥。可以看到v14-v10这里组成了个数组。我们将其取出。秘钥即为 `thisisatestkey==`

然后自己构造一个AES脚本对我们之前拿到的数据进行解密。

解题-踩坑ing..

解密得到aimagetencent。提交flag，发现失败，猜测这个即为我们的登陆账号密码。 然后安装apk，尝试登陆发现闪退。



我们只能继续去看一下登陆逻辑，没发现什么。（其实是跟不下去了

然后找了一下大佬们的wp。发现存在一个没有被调用过的FileDataActivity。 flag在这里。

于是拿到我们的字符串，然后解密，即可得到flag。

解密脚本如下：

```
#coding:utf-8
import base64
from Crypto.Cipher import AES

class AesEncry(object):
    key = "thisisatestkey=="  # aes秘钥

    def encrypt(self, data):
        data = json.dumps(data)
        mode = AES.MODE_ECB
        padding = lambda s: s + (16 - len(s) % 16) * chr(16 - len(s) % 16)
        cryptos = AES.new(self.key, mode)
        cipher_text = cryptos.encrypt(padding(data).encode("utf-8"))
        return base64.b64encode(cipher_text).decode("utf-8")

    def decrypt(self, data):
        cryptos = AES.new(self.key, AES.MODE_ECB)
        decrpytBytes = base64.b64decode(data)
        meg = cryptos.decrypt(decrpytBytes).decode('utf-8')
        return meg[:-ord(meg[-1])]


a =AesEncry().decrypt("9YuQ2dk8CSaCe7DTAmaqAA==")
print(a)
```