记一次APP测试的爬坑经历

E安全 今天

以下文章来源于雷神众测 ， 作者Z1ng3r22

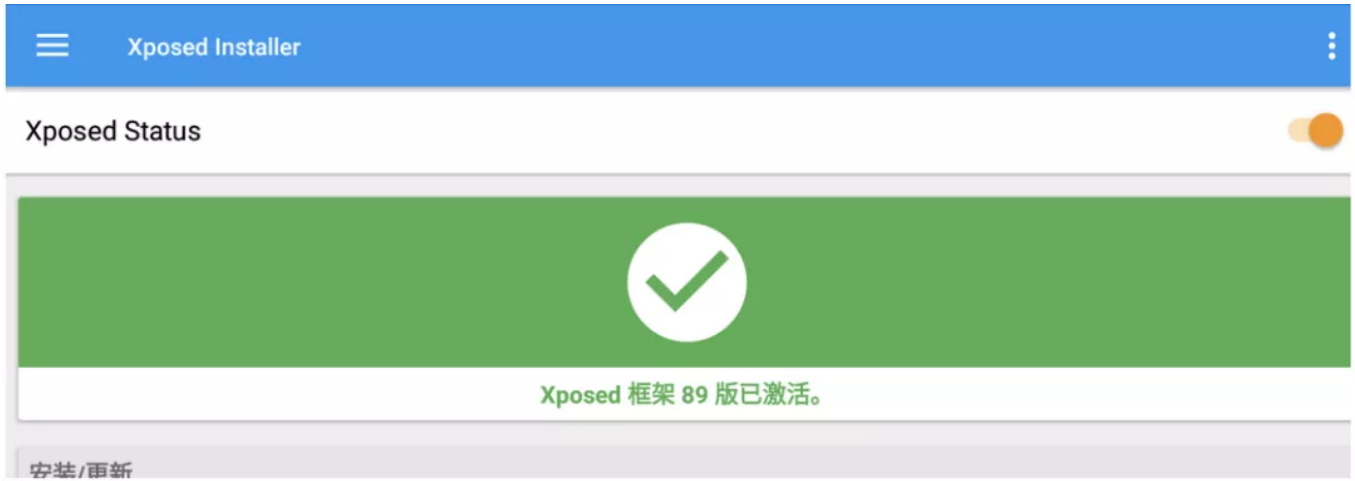
声明

No.1 问题—闪退

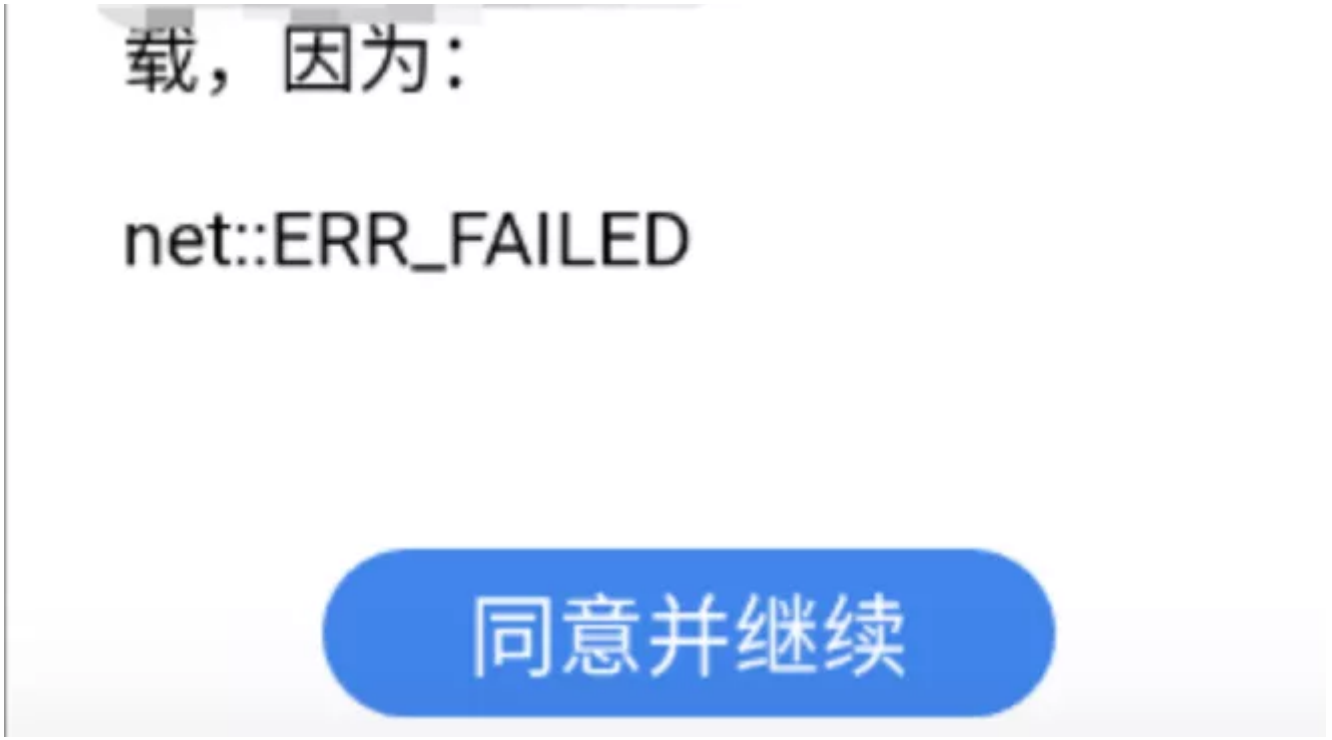
开始在接到任务的时候，由于是第一次接触APP测试，兴高采烈的将其装进夜神模拟器里面，准备学习一番。结果，轻触APP，它轻轻地来了，却又轻轻地走了（对没错，它无脑闪退。。。）
随后各位大佬告诉我，可以更换真机、模拟器、安卓版本各种尝试。
在经过一系列的测试后，将夜神模拟器的版本换为 Android 7 成功解决闪退问题

No.2 抓包

为了抓取APP的数据包，我将burp的证书装入我的模拟器中。但是，却一直抓不到，在做了一定功课了解后，知道了可以采用 Xposed + JustTrustMe 来突破 SSL PINNING 。



但是！模拟器在未装 JustTrustMe 时候无法抓包，而在装上以后，整个网络环境，不可用。打扰了~



还好，感谢阿雄送的 JustTrustMePlus （其只令某一个APP强制信任证书）。

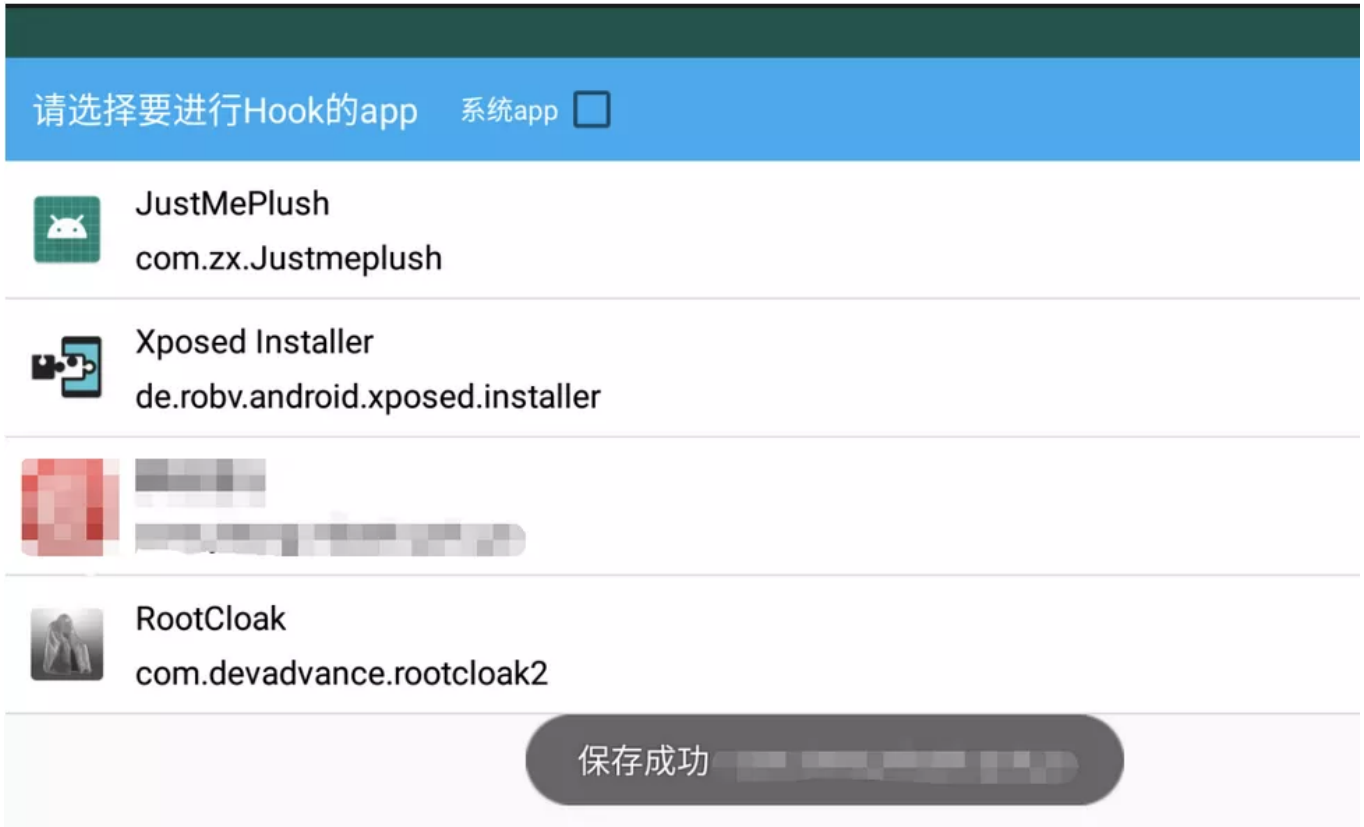
好像是高版本里面本身和justtrustme冲突

我这里也是一样的

安卓高版本的原因么?

已读

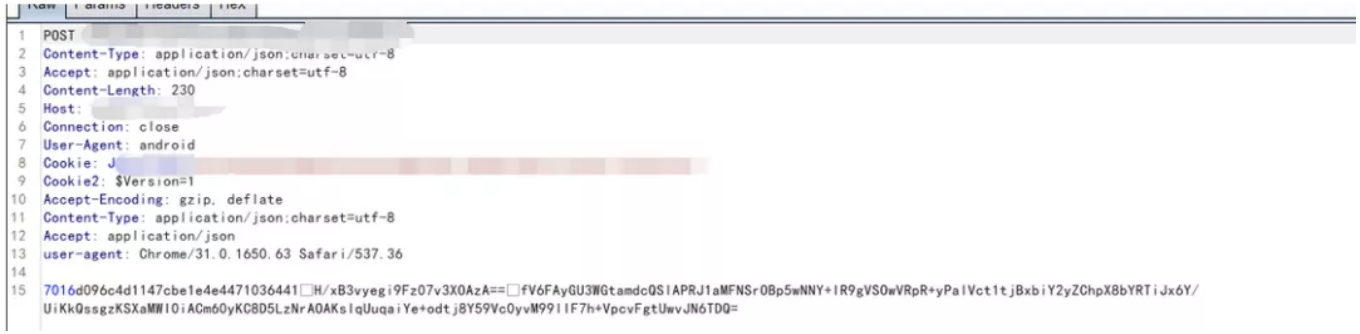
嗯



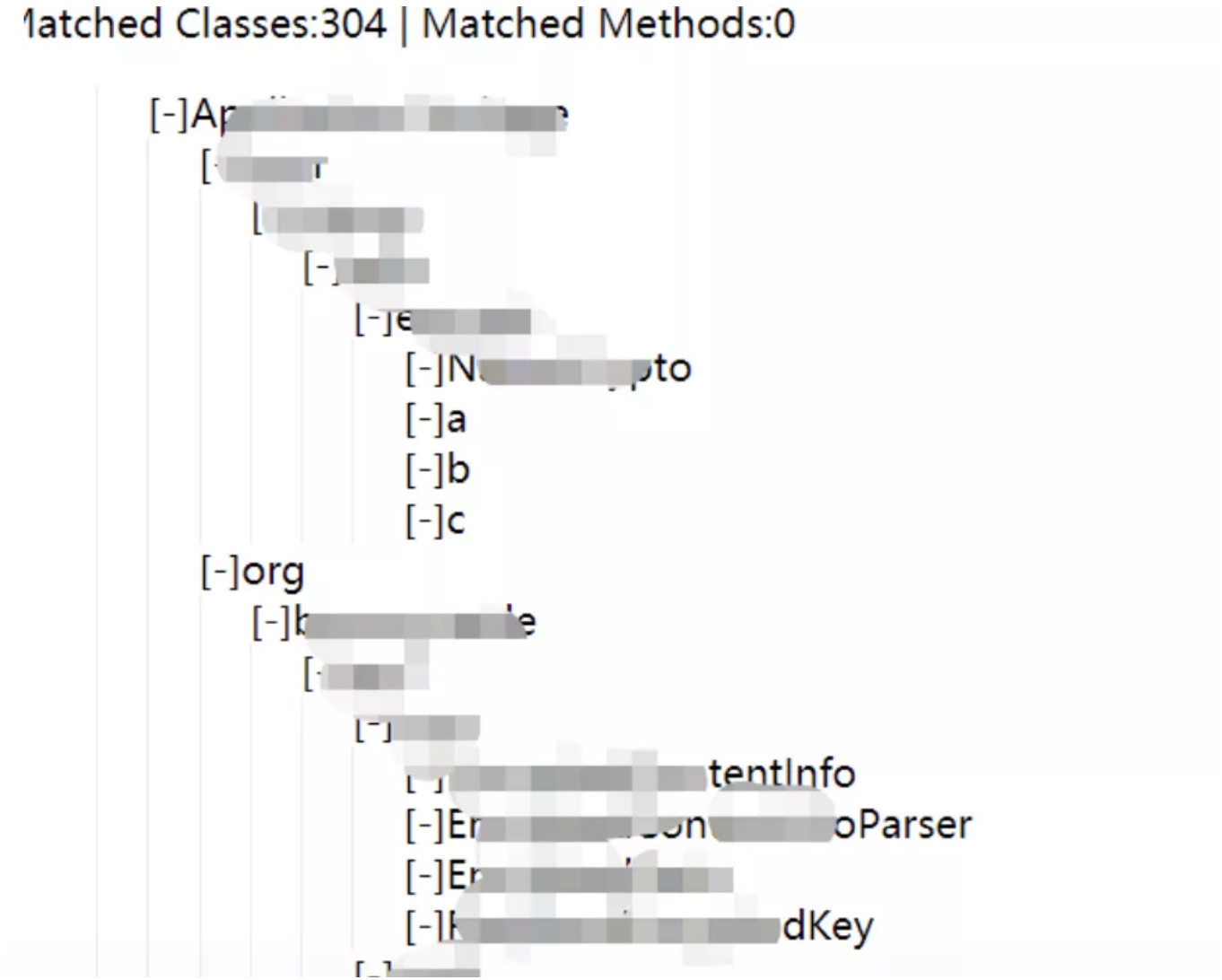
再用 Burp 进行抓包的时候，就能成功抓取到APP的数据包了。

No.3 传输加解密

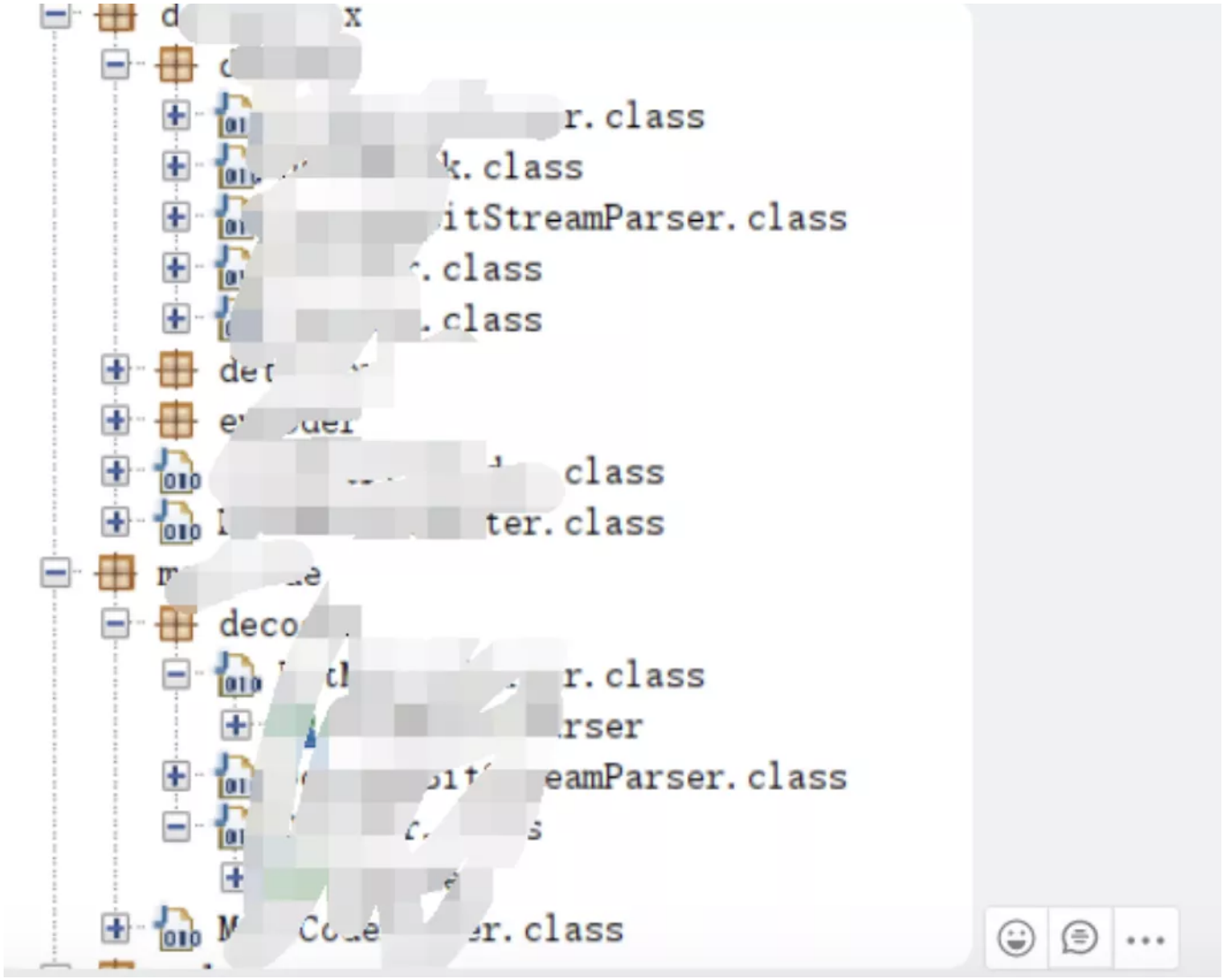
在进行抓包测试的时候，发现传输过程存在加解密。



起初，使用阿雄教的方法，用 Xserver 去动态 hook 加解密（需要知道其加解密函数），可是却没有任何反应。



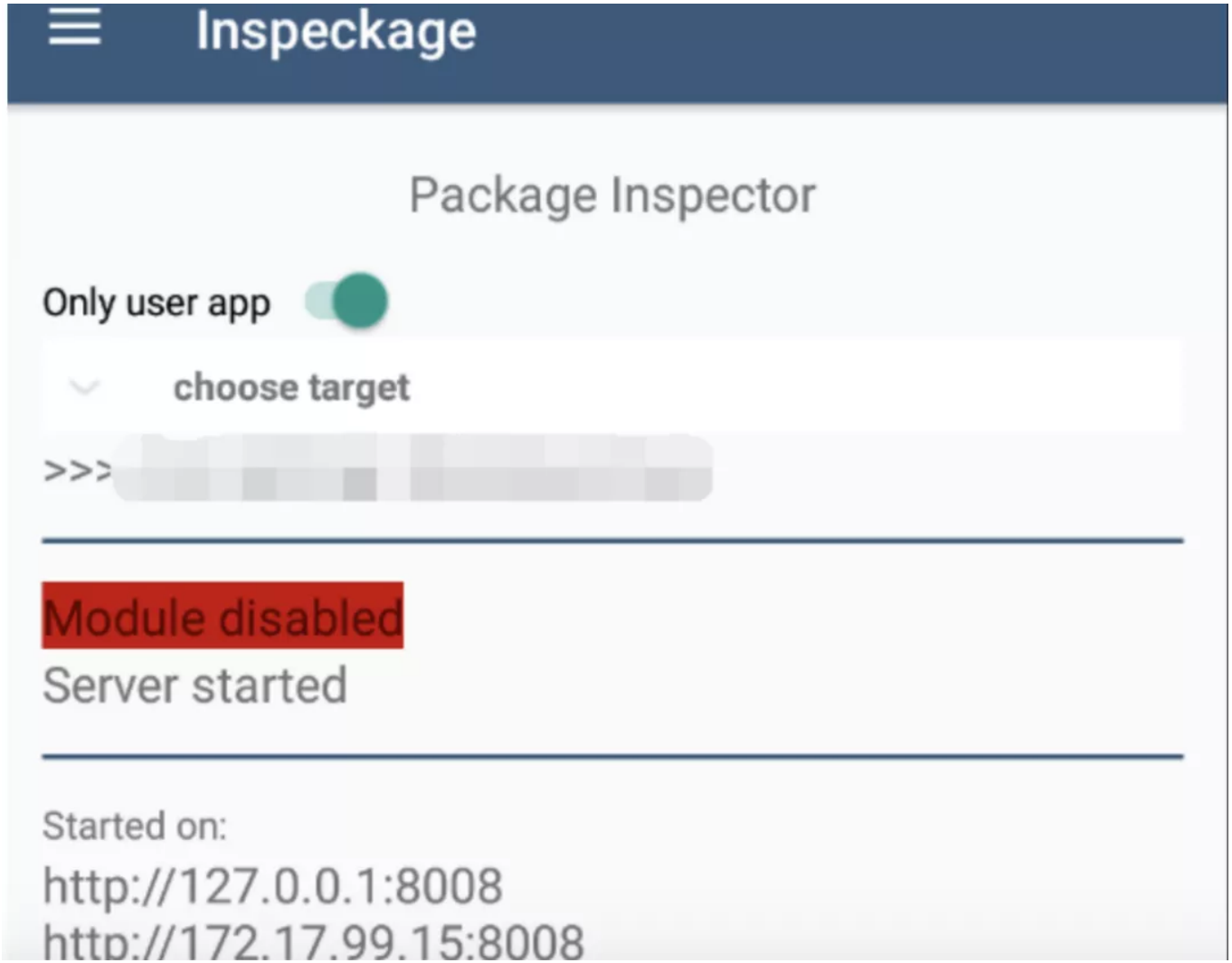
无能为力的我，便去求同事帮忙逆向一下，却是无果



Inspeckage：是一个用于提供帮助Android应用程序动态分析的工具。通过对Android API的函数使用hook技术，帮助用户了解应用程序在运行时的行为。Gayhub地址：https://github.com/ac-pm/Inspeckage

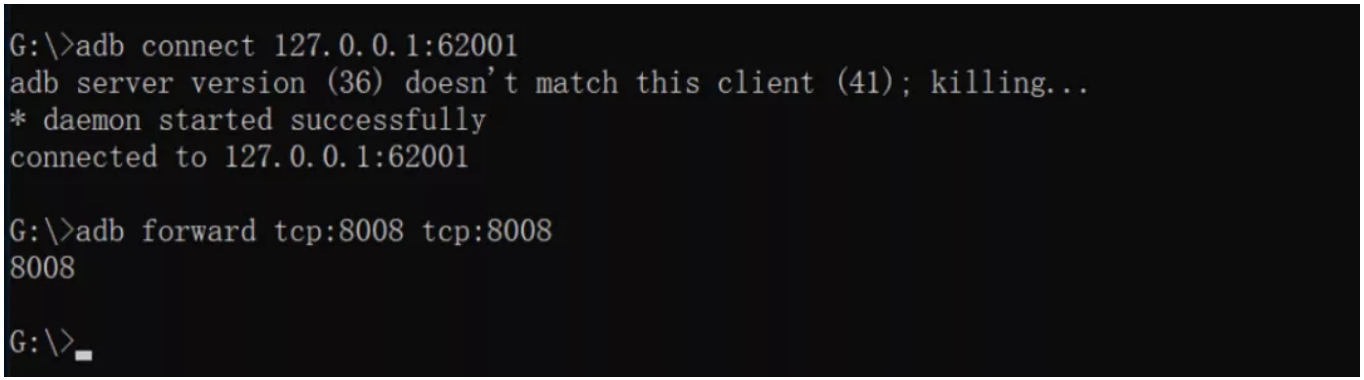
操作如下：

1、勾选需要调试的APP



2、本机进行端口转发

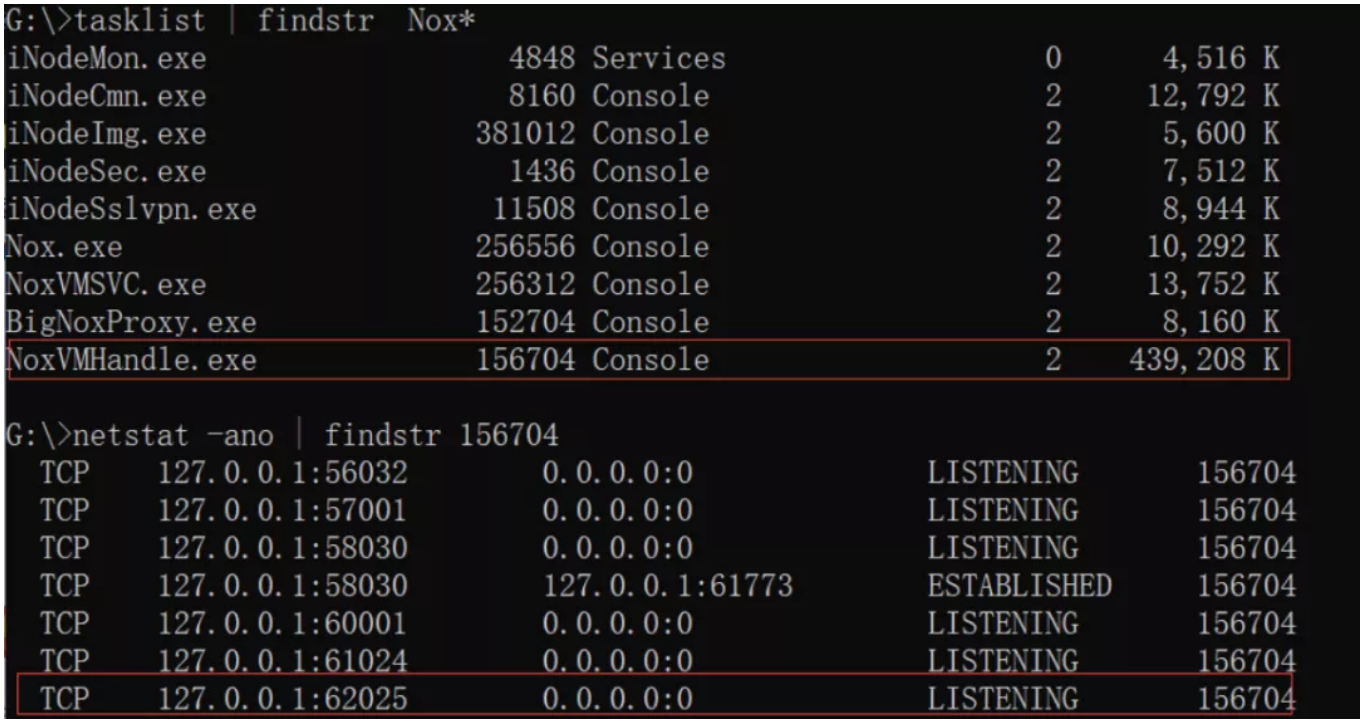
```
adb connect 127.0.0.1:62001
adb forward tcp:8008 tcp:8008
```



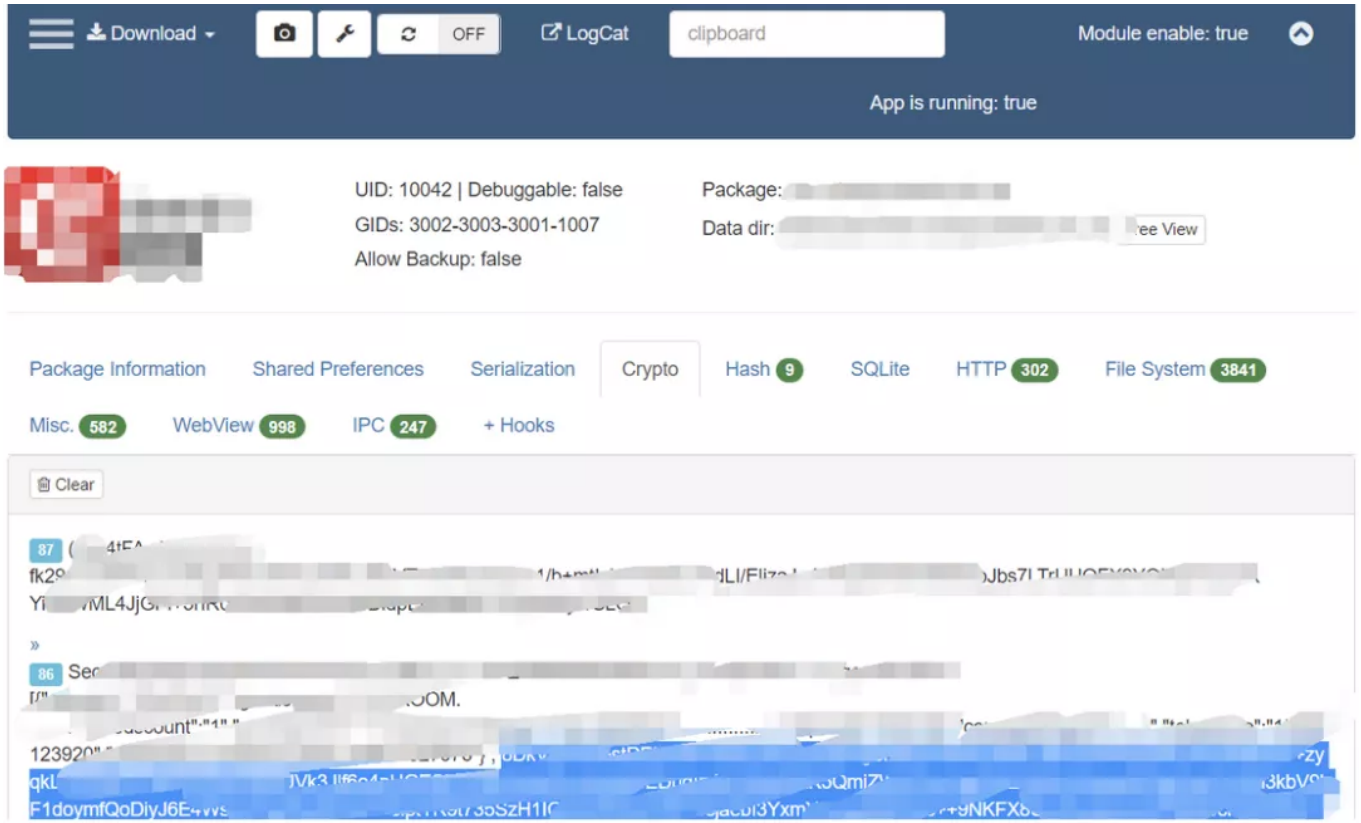
Tips:
62001是夜神模拟器在本机的默认端口
而关于模拟器多开的问题，端口计算公式如下：

```
nox_i = 62024+i
例如：
nox_1 = 62025
nox_2 = 62026
nox_3 = 62027
```

当然，也可以通过命令行查看端口



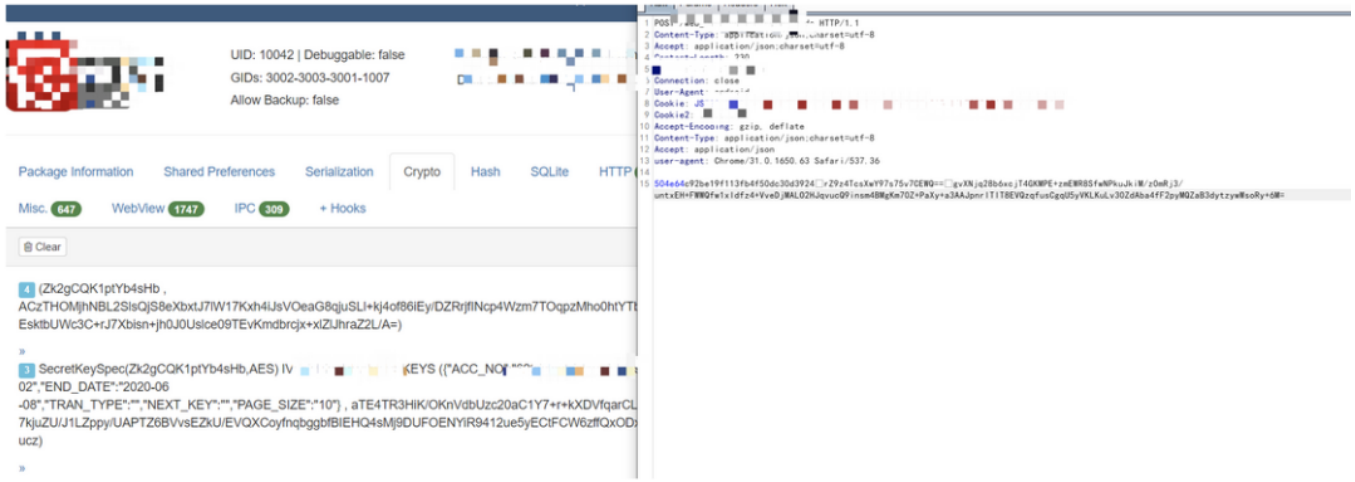
3、打开Inspeckage



将上方的 OFF 勾选为 ON 就能成功 hook 函数了

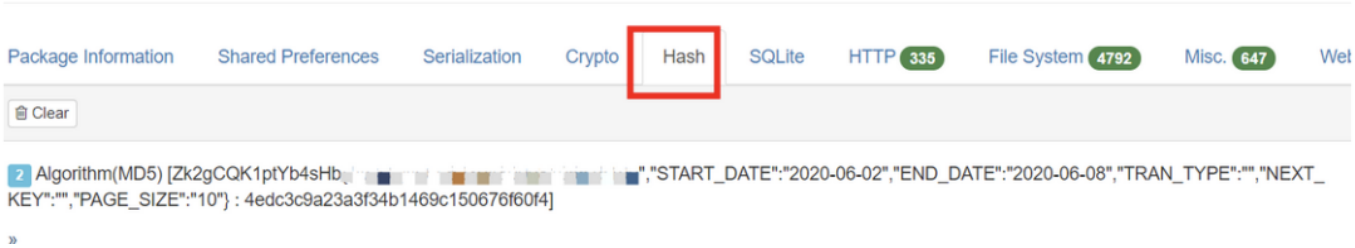
4、配合Burp分析加解密

在抓取数据包后，我发现，POST的数据使用 0x1d 作为分隔符将其分为三段

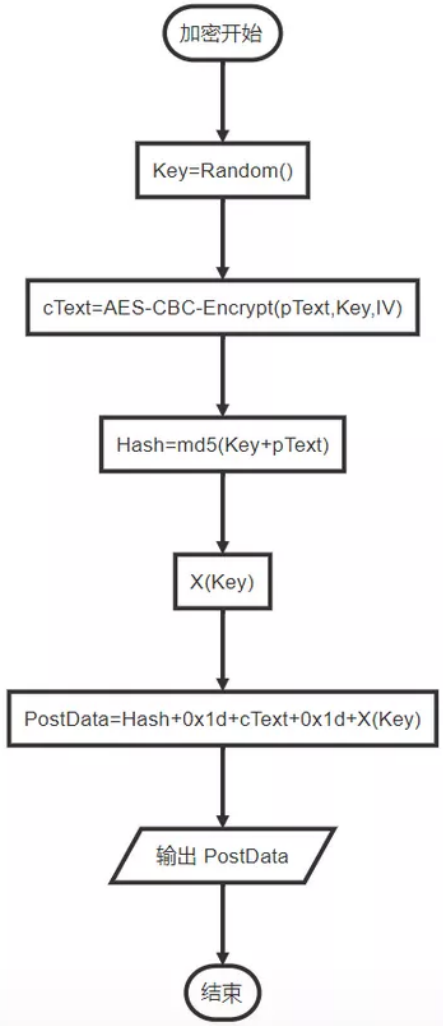


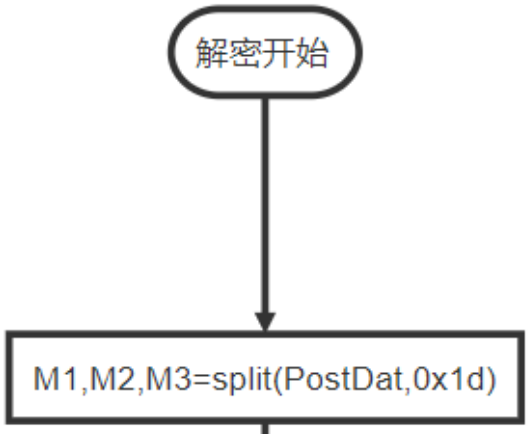
多抓了几个数据包后，综合分析出，BurpSuite截获的POST数据包被 0x1d 字符分成了三段M1,M2M,M3，其中：
M2的动态生成过程对应 Inspeckage 中的 3，其使用了 AES CBC 模式进行加解密，且 IV 初始偏移量 为一个固定值（Inspecakge中能看到）。
M3可以看到通过 X 算法加密前，恰好为第二段的密钥（第三段的密钥是每次请求都会动态生成的，我并没有解密出第三段的加密方法，只能每次使用Inspeckage查看密钥，如果有大佬了解的，请不吝赐教）

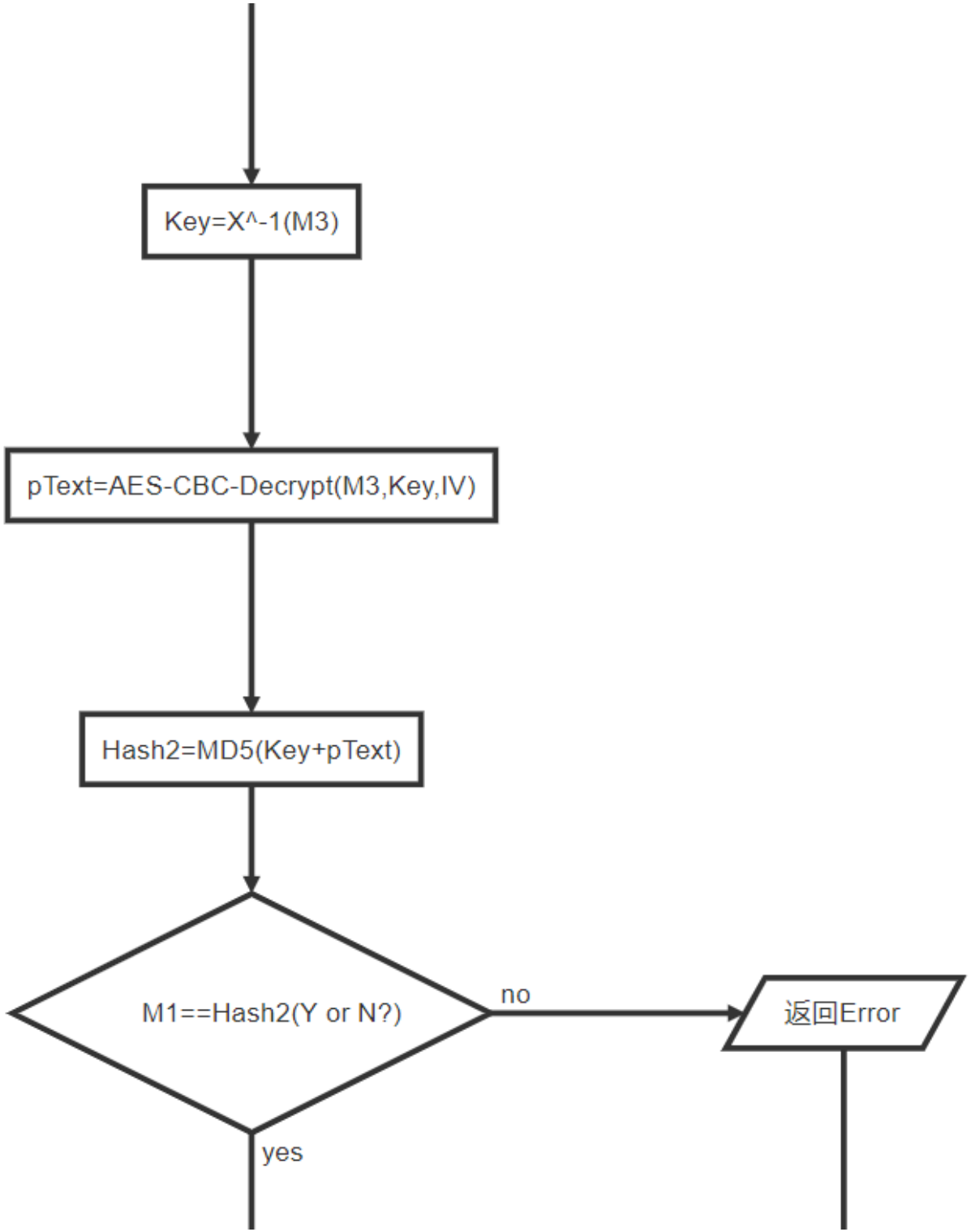
而M1，使用了密钥+明文再经过一次 MD5 散列的处理，在后端做篡改的校验

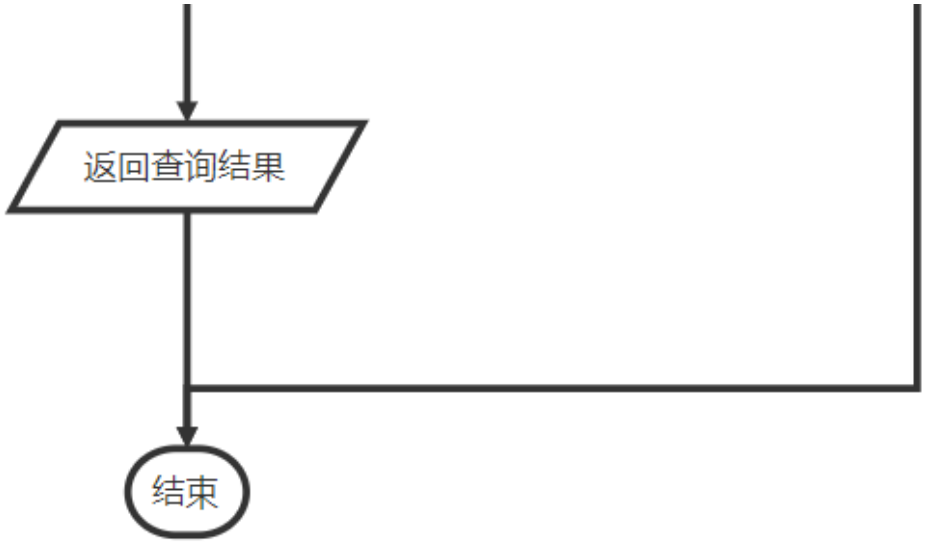


流程图如下：


加密开始
Key=Random()
cText=AES-CBC-Encrypt(pText,Key,IV)
Hash=md5(Key+pText)
X(Key)
PostData=Hash+0x1d+cText+0x1d+X(Key)
输出 PostData
结束


解密开始
M1,M2,M3=split(PostDat,0x1d)


Key=X^-1(M3)
pText=AES-CBC-Decrypt(M3,Key,IV)
Hash2=MD5(Key+pText)
M1==Hash2(Y or N?)
no
返回Error
yes

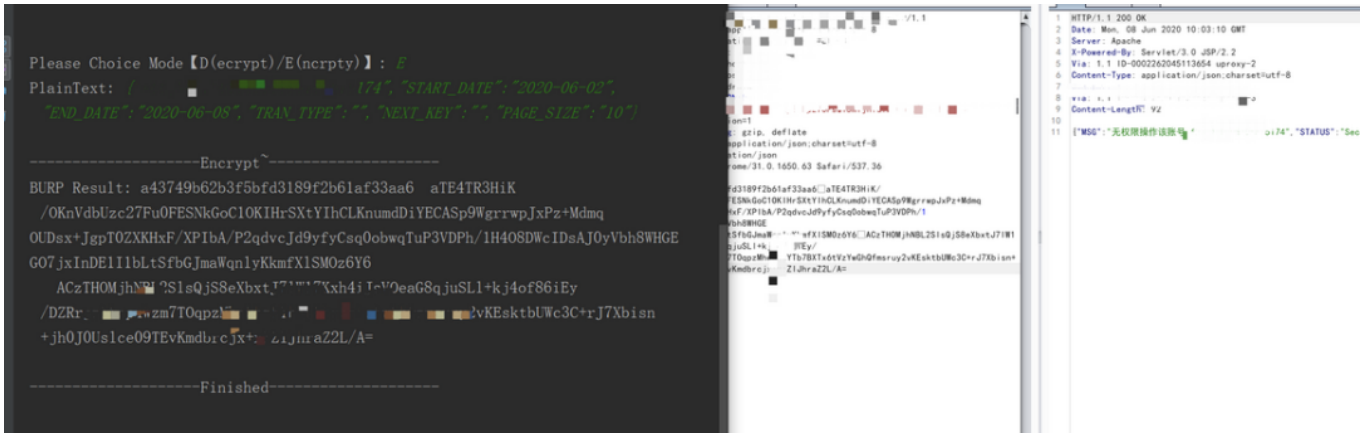


5、编写小脚本

分析出结果后，便能根据思路写一个脚本出来简化操作了

解密测试：

修改参数再加密的测试（此处进行越权测试）：



No.4 后续思考

在后续进行测试的时候，由于每次请求都需要把密钥和密钥解密前的密文输入程序，导致整个测试流程较为繁琐，进度较慢。（由于不会写Burp插件，手法比较笨）

于是回想了下整个加解密流程，发现关键是：

1、密钥 Key1 是本地通过代码随机生成的，经过加密流程生成 PostData 。

2、服务器端经过代码解密出Key2 ，不和客户端生成的 Key1 做校验，只要能成功解密出数据就行。

也就是说，我们可以将 Key2 写为固定值，然后每次只需要输入需要加密的数据就行了。当然，解密过程还是需要我们输入 Key1 的。

于是我取了某次在 Inspeckage 中取到的 Key 和 X(Key) 写为定值，修改了原本的代码，简化了操作。

```
package com.encrypt;

import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

import sun.misc.BASE64Encoder;
import sun.misc.BASE64Decoder;
import org.apache.commons.codec.digest.DigestUtils;

import java.util.Arrays;
import java.util.List;
import java.util.Scanner;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
```

```
class AesCBC {

    //IV
    private static String InitV="XXXXXXXXXXXXXXXX";

    private String D="";

    /**
     * FUNCTION: ENCRYPT
     * */
    public String encrypt() throws Exception{
        Cipher cipher=Cipher.getInstance("AES/CBC/PKCS5Padding");
        SecretKeySpec sks=new SecretKeySpec(getKey().getBytes(),"AES");
        IvParameterSpec iv=new IvParameterSpec(InitV.getBytes());
        cipher.init(Cipher.ENCRYPT_MODE,sks,iv);
        byte[] ctext=cipher.doFinal(getE().getBytes());
        return new BASE64Encoder().encode(ctext);
    }

    /**
     * FUNCTION: DECRYPT
     * */
    public String decrypt() throws Exception{
        Cipher cipher=Cipher.getInstance("AES/CBC/PKCS5Padding");
        SecretKeySpec sks=new SecretKeySpec(getKey().getBytes(),"AES");
        IvParameterSpec iv=new IvParameterSpec(InitV.getBytes());
        cipher.init(Cipher.DECRYPT_MODE,sks,iv);
        byte[] ptext=cipher.doFinal(new BASE64Decoder().decodeBuffer(getD()));
        return new String(ptext);
    }


    public static void main(String[] args) throws Exception {
        //开始
        System.out.println("--------------------~Strat~--------------------");
        System.out.println("************************************************");
        AesCBC ac=new AesCBC();
        Scanner sc = new Scanner(System.in);

        while(true){
            System.out.print("Please Choice Mode[D(ecrypt)/E(ncrpty)]: ");
```

```
            String mode=sc.nextLine().toUpperCase().trim();

            if(mode.equals("D")){
                //-------解密-------
                //输入key
                System.out.println("--------------------AESKey~--------------------");
                System.out.print("Key: ");
                String key1=sc.nextLine().trim();
                ac.setKey(key1);
                System.out.print("CipherText: ");
                String D=sc.nextLine().trim();
                ac.setD(D);
                System.out.println("\n--------------------Decrypt~--------------------");
                String ptext=ac.decrypt();
                System.out.println("Decrypt Result: "+ptext);
                System.out.println("\n--------------------Finished--------------------\n\n");
            }else if(mode.equals("E")){
                //-------加密-------
                //key为定值
                System.out.println("--------------------AESKey~--------------------");
                String K="(HqttsSdHpJHTwkF7 , WUy7AghAGVmCHcdC78jW+wTbCi2SvJ7n3Ig6Mmbi+Qdkn5TL7
9ISZ8XnIA03KRDhtZmPltbJSCQaIw8TbkkzY7wK/SpUmK0+wZV4feYDf+4RIAHCQyA+bWXx1dvdJT00toNyrQSCuORsCh2V
MusmJ6XhyI1MrYNDY3m+1poNqes=)";
                System.out.println("HqttsSdHpJHTwkF7\n");
                String str="";
                //正则匹配括号里的，并用逗号分割
                Pattern pattern = Pattern.compile("(?<=\\()[^\\)]+");
                Matcher matcher = pattern.matcher(K);
                while(matcher.find()){
                    str=matcher.group();
                }
                List<String> keys = Arrays.asList(str.split(" , "));
                String key1=keys.get(0);
                String key2=keys.get(1);
                ac.setKey(key1);
                System.out.print("PlainText: ");
                String E=sc.nextLine().trim();
                ac.setE(E);
                System.out.println("\n--------------------Encrypt~--------------------");
                String ctext=ac.encrypt();
```

```
            String hashStr=ac.getKey()+ac.getE();
            String hashedStr=DigestUtils.md5Hex(hashStr);
            char sp=29;
              System.out.println("BURP Result: "+ hashedStr+sp+ctext+sp+key2);
            System.out.println("\n-------------------Finished-------------------\n\n");
        }else if(mode.equals("0")){
            System.out.println("\n*********************************************");
            System.out.println("-------------------the End-------------------");
            System.exit(0);
        }

    }
```

No.5 总结

通过本次APP的测试，总的来说是艰辛却收获了很多，但最后就结果来讲有两点不太完美的地方

1、对于M3并未成功破解，否则可以写出更加简化操作的代码

2、由于不会写Burp插件，所以操作过程还是比较繁琐（下次一定学着写💪💪）

> 注：本文转载自公众号雷神众测 https://www.easyaq.com