hook从未如此丝滑

原创海岸线突击队 酒仙桥六号部队
2020-12-29原文

这是 **酒仙桥六号部队** 的第 **138** 篇文章。
全文共计2143个字，预计阅读时长7分钟。

朋友们好啊

我是葫芦兄弟掌门人铁头娃！

刚才有个朋友问我：“铁老师，发生甚么事了？”

我说怎么回事？

给我发了几张截图，我一看！

奥！原来是昨天有个app，54多兆。

不仅有壳抓包还加密。

塔门说：

有一个说是我对这个app做测试，脱壳脱的头发都没了。

铁老师你能不能教教我简单的测法？

帮我缓解下我的工作量。

我说可以~

我说你一个一个脱壳练死劲不好用，他不服气。

我说小朋友，你用脱壳来比我hook。

他说比不过，

他说你这个没用，

我说我这个有用。

这是化劲，app测试是讲化劲的，四两拨千斤。

他非要和我试试！

我啪一下就站起来，上来就先搭建环境

本次测试采用模拟器环境。

frida环境搭建略过，*度资料很多。

使用的是mumu模拟器（adb连接比*神模拟器方便很多）。

以及httpdecrypt

（https://github.com/lyxhh/lxhToolHTTPDecrypt）

首先连接adb并运行frida-server；

```
C:\Users\xxx>adb connect 127.0.0.1:7555
connected to 127.0.0.1:7555

C:\Users\xxx>adb shell
root@x86:/ #
```

启动httpdecrypt；

```
python3 app.py
```

在运行时可能遇到报错；

```
[ERROR] device not found, please wait for few seconds and retry.
```

解决：全局搜索get_usb_device，修改为get_usb_device（timeout=1000）即可。

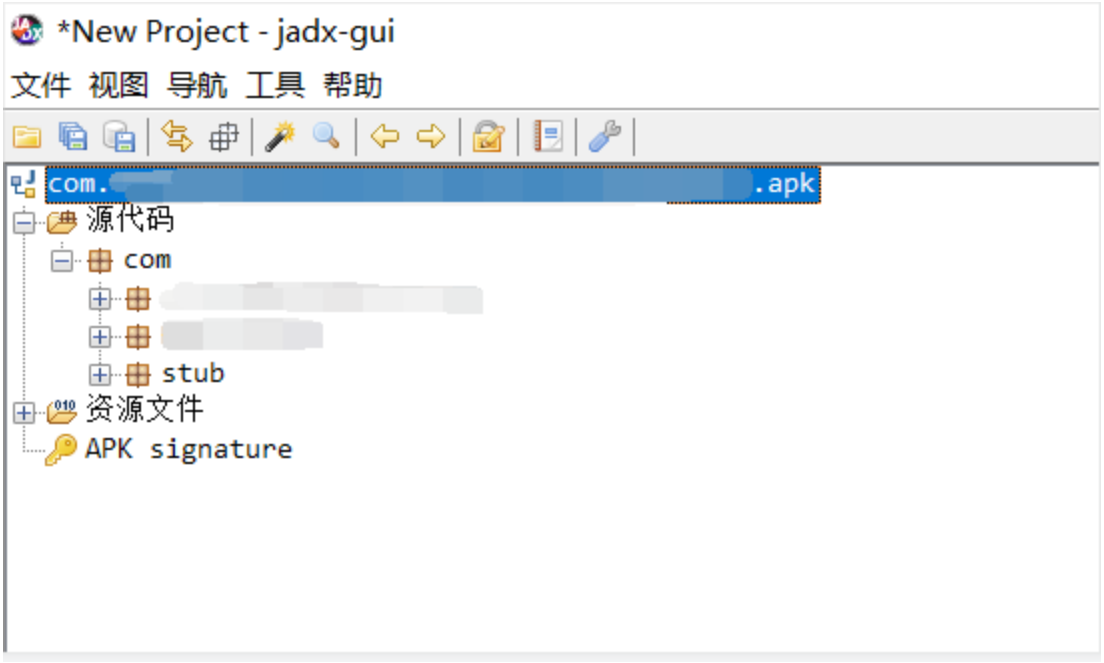
下载burp插件HTTPDecrpyt并进行安装。https://github.com/lyxhh/lxhToolHTTPDecrypt/releases

这个app不讲武德

首先正常对app抓包；

可以看到app对所有参数都进行了加密，不论是请求包还是返回包；

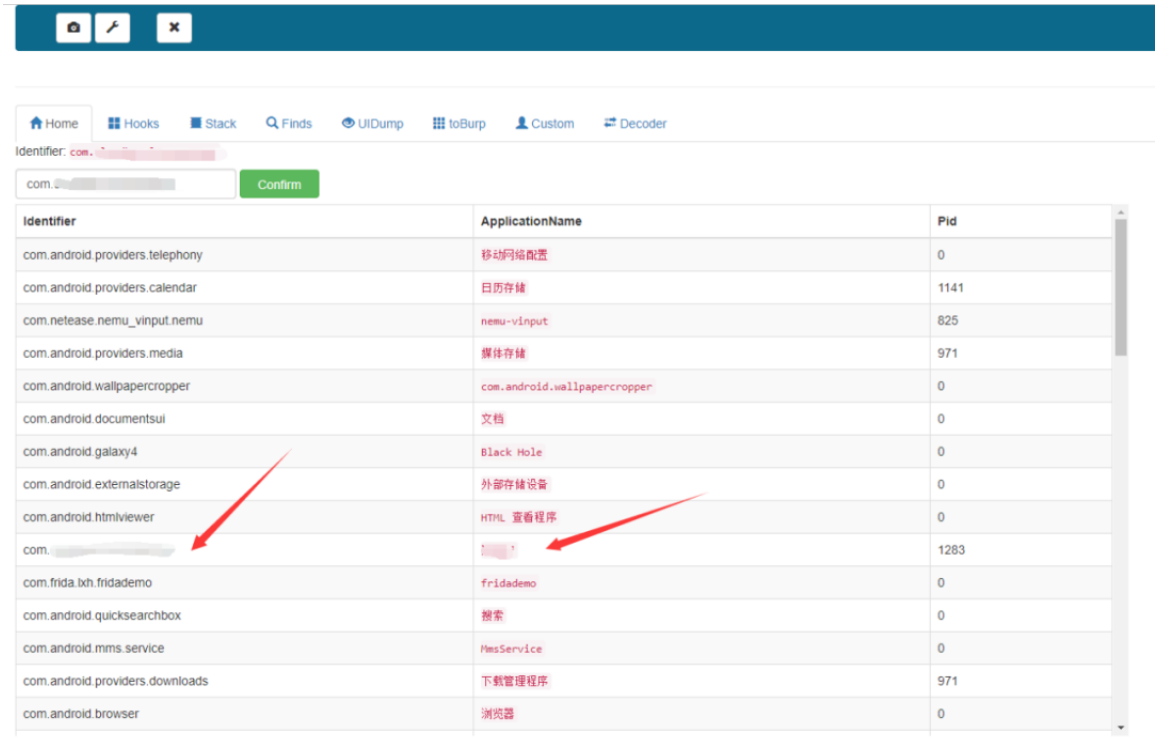
用神器jadx对app进行逆向试试；



有壳，还有加密。

他上来就是一个加壳，一个加密，我全都防出去了嗷

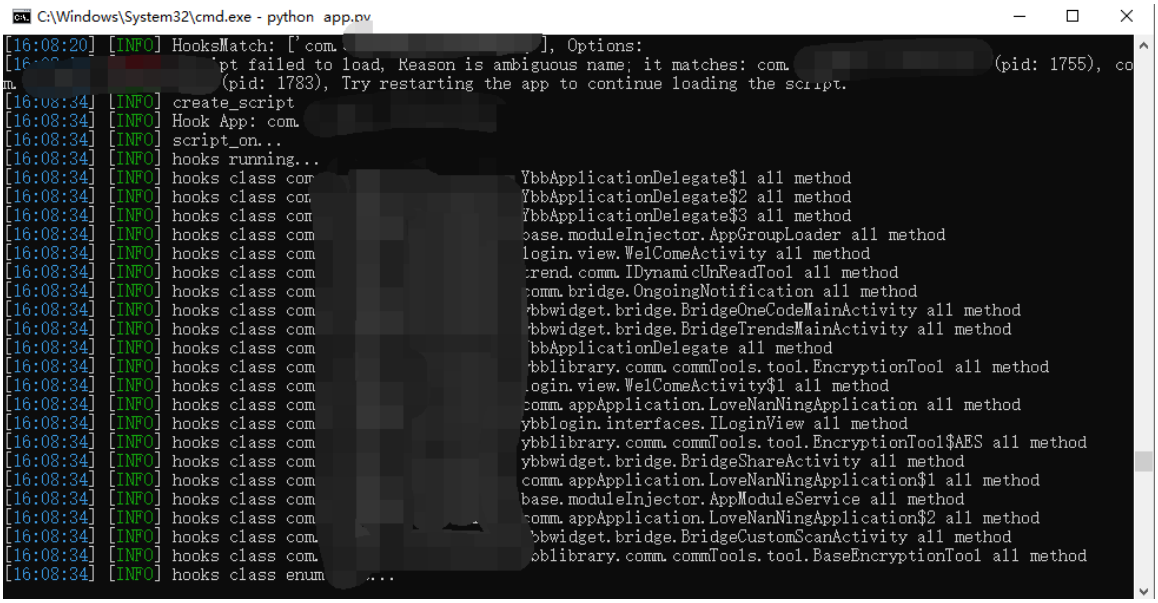
浏览器访问http://127.0.0.1:8088/



找到目标app，将app包名com.xxx.xxx填入框中并点击Confirm；

然后点击Hooks功能；

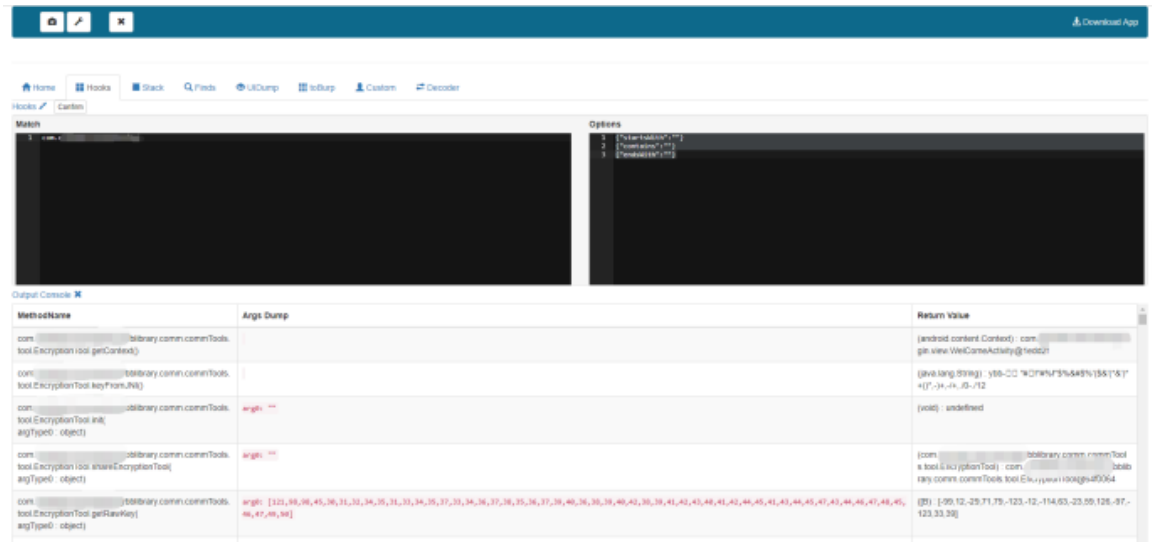
在Match中输入刚才的包名，点击Confirm。



等待输出：

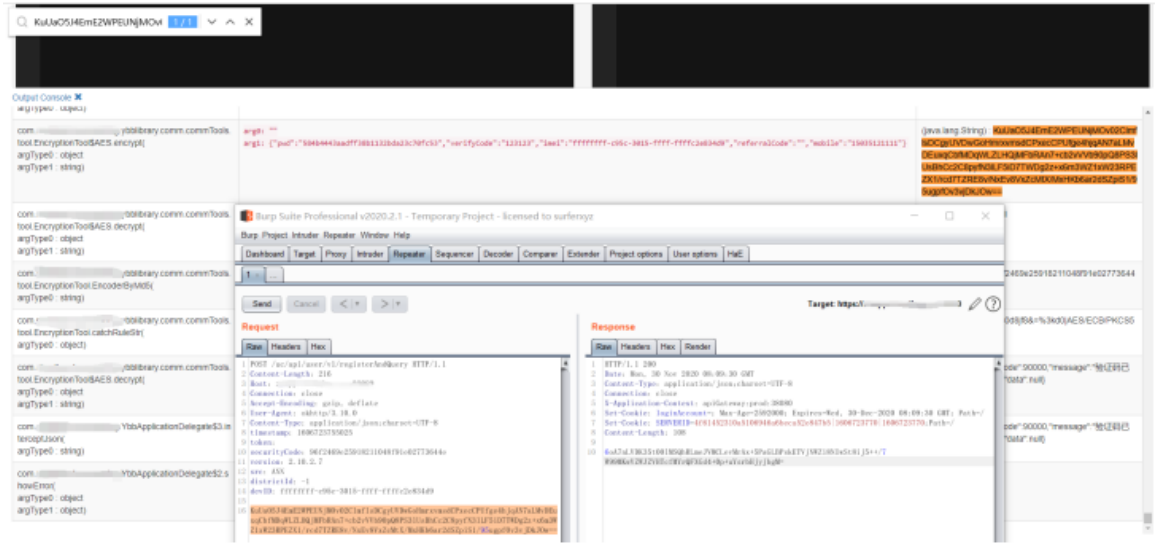
```
[INFO] hooks class enum done...
```

此时，已经将该app所有方法进行了hook。



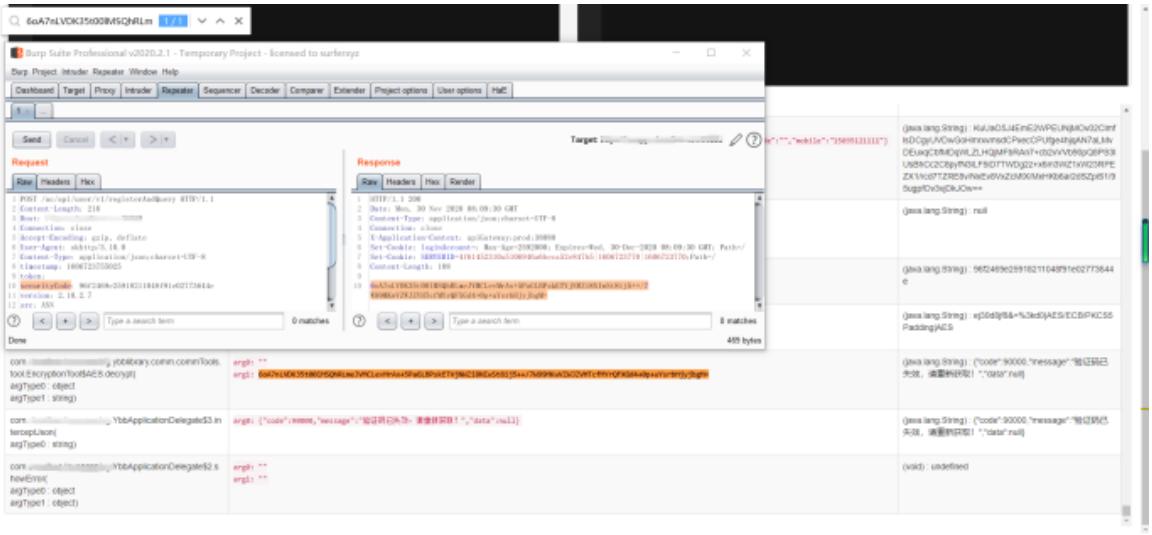
我们重复刚才的操作。

根据数据包中加密后的数据，找到相应的方法。



```
com.xxx.xxx.ybblibrary.comm.commTools.tool.EncryptionTool$AES.en
crypt(argType0 : object argType1 : string)
```

可以看到是这个函数将请求包的数据进行了加密。



com.xxx.xxx.ybblibrary.comm.commTools.tool.EncryptionTool$AES.decrypt(argType0 : object argType1 : string)

是返回包的解密函数。

我们看下加密函数：

com.xxx.xxx.ybblibrary.comm.commTools.tool.EncryptionTool$AES.encrypt(argType0 : object argType1 : string)

可以看到加密函数需要两个参数，分别是对象和字符串。

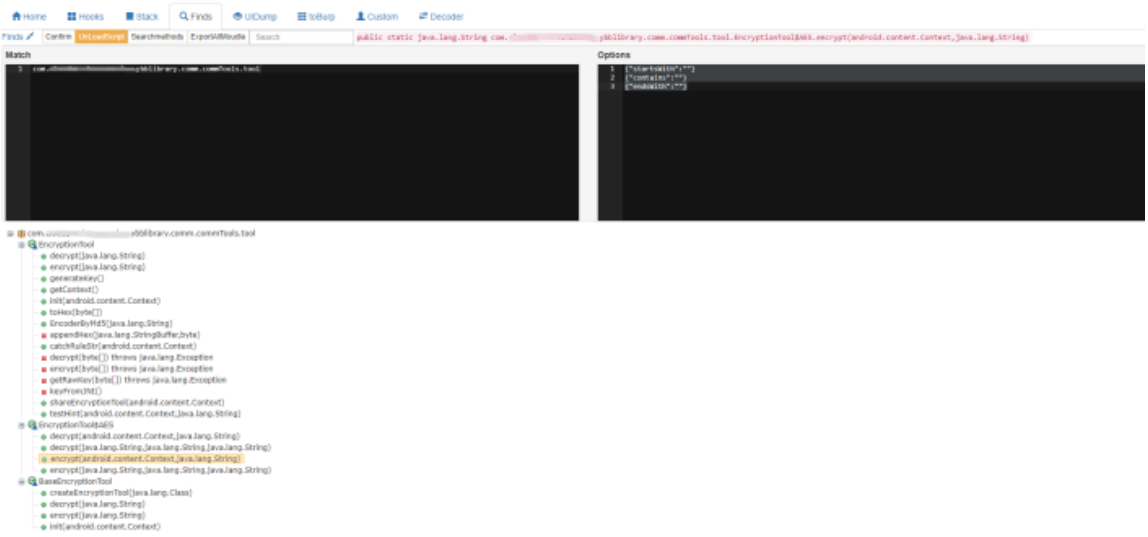
但是我们需要知道对象参数的具体类型。

现在转到Finds功能中，搜索函数的类名。

com.xxx.xxx.ybblibrary.comm.commTools.tool

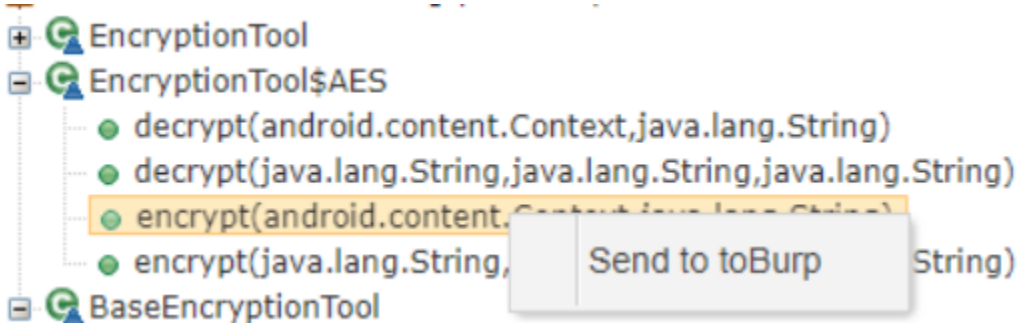
点击Confirm。

找到方法名EncryptionTool$AES.encrypt。



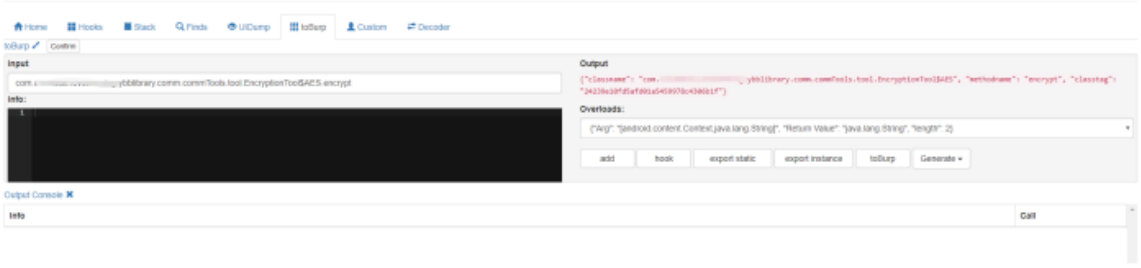
可以看到加密函数参数中，对象类型的参数具体类型为static。

选择到encrypt，然后右键发送到toBurp功能。



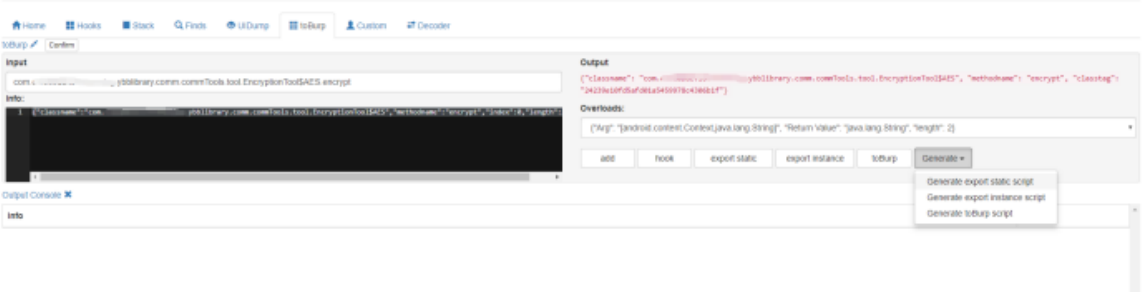
转到toBurp功能。

点击Confirm。



点击add。

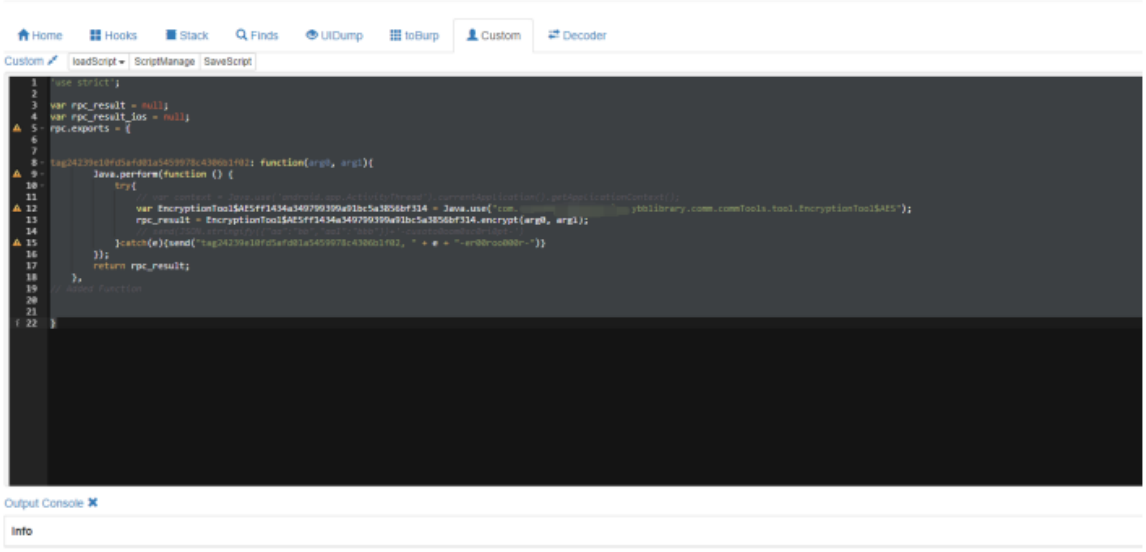
由于这个加密函数有两个参数，所以需要自己编辑代码（默认为一个参数）。



由于对象参数类型为static，所以我们选择Generate export static script。

如果参数为动态，那我们选择Generate export instance script。

现在可以看到，在custom中会生成代码。



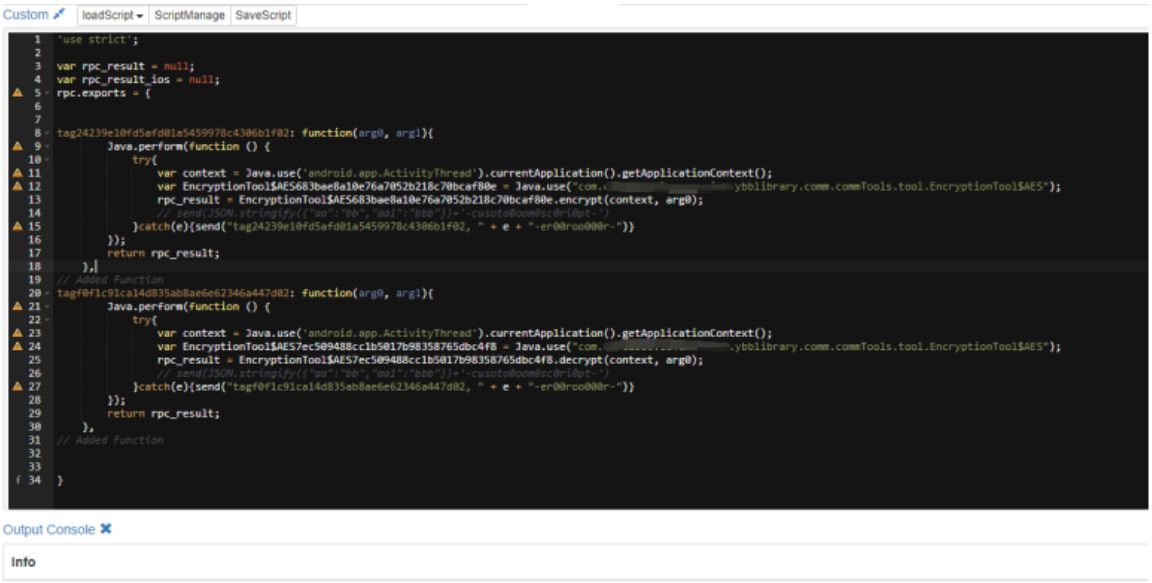
代码中，arg0是传过来的加密字符串，arg1是Android的Context对象。

添加代码：

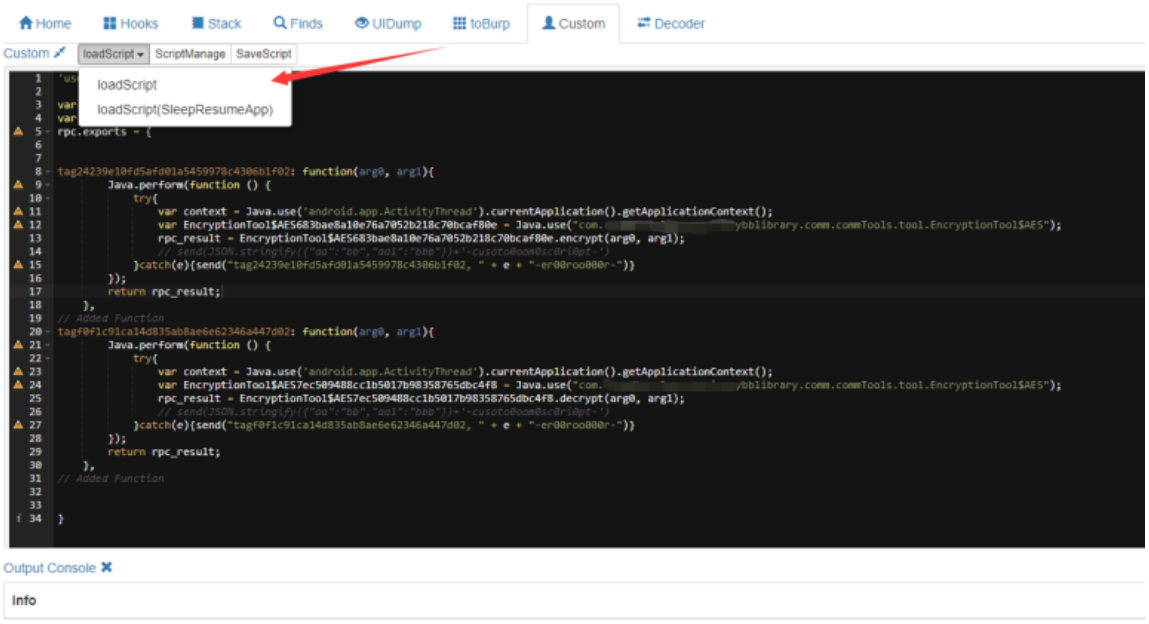
```
var context =
Java.use('android.app.ActivityThread').currentApplication().getA
pplicationContext();
```

再用同样的方法生成decrypt的代码。

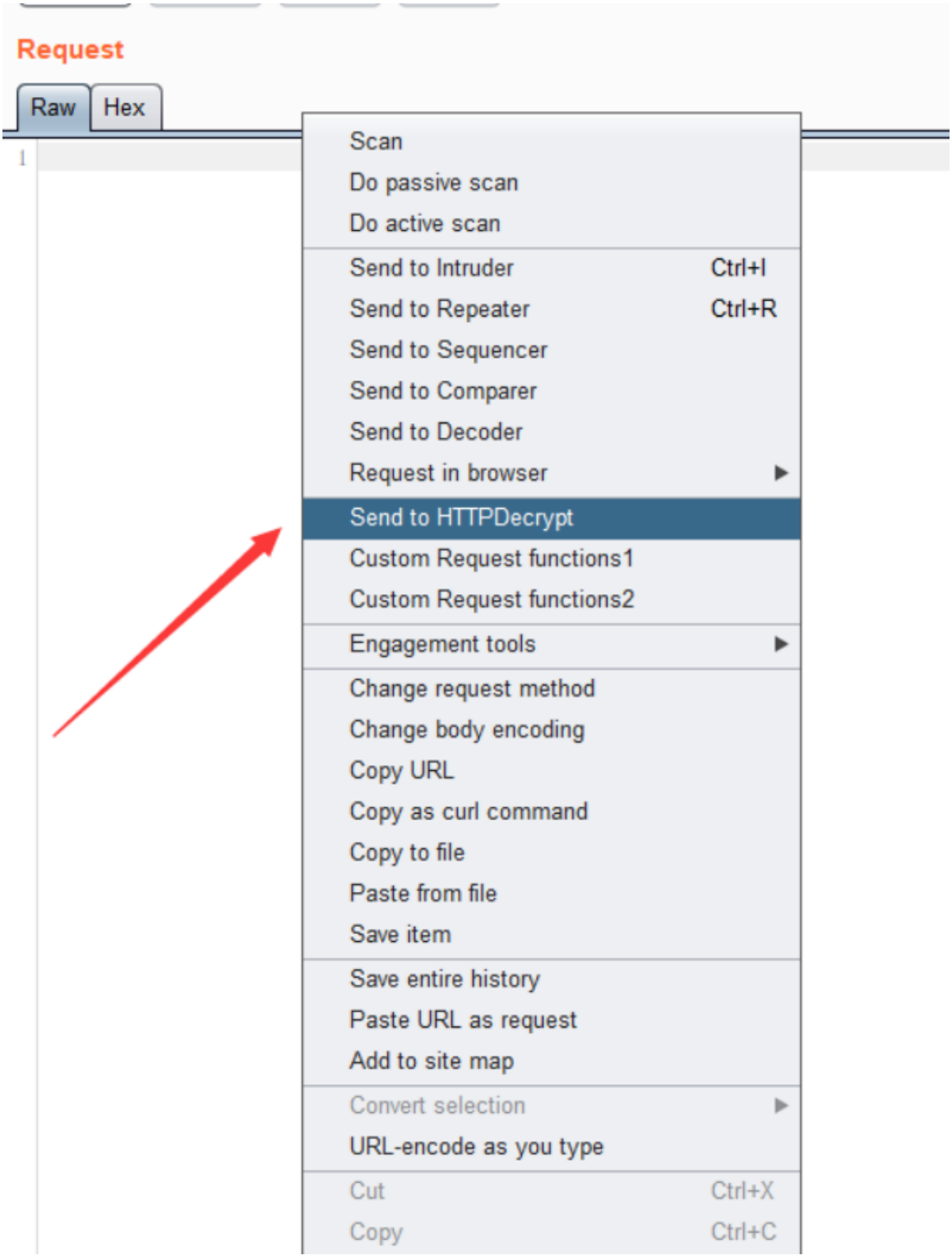
最终代码为：



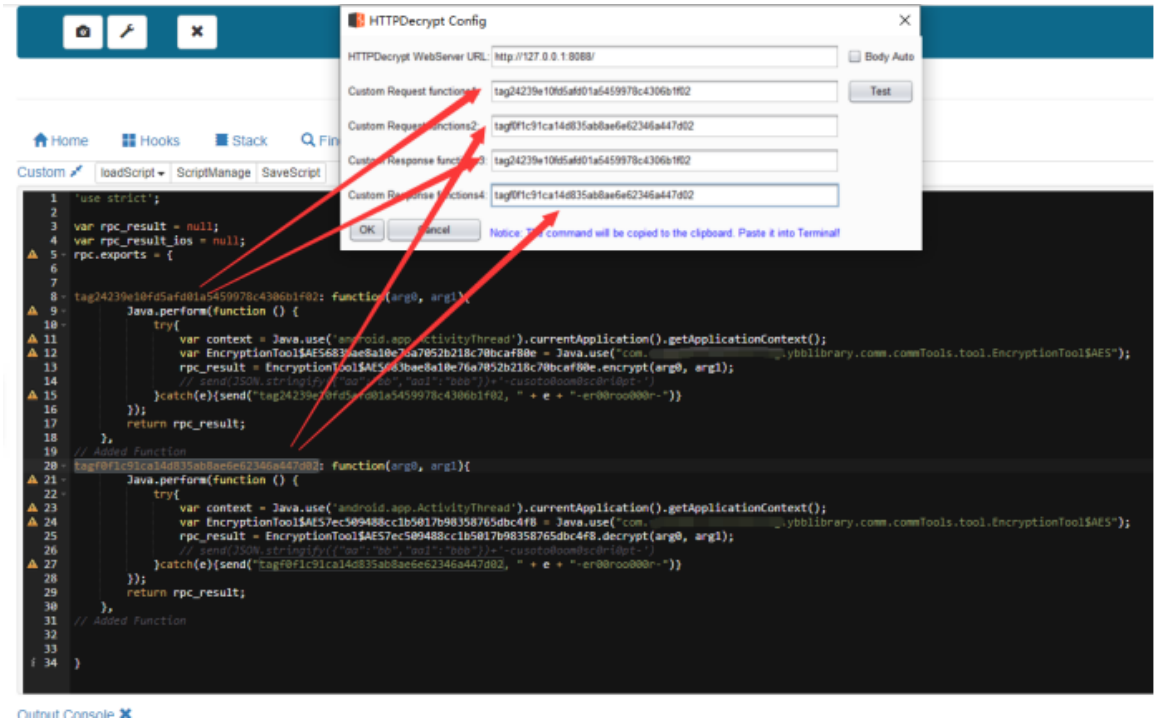
点击左上角loadScript 将脚本发送到burp。



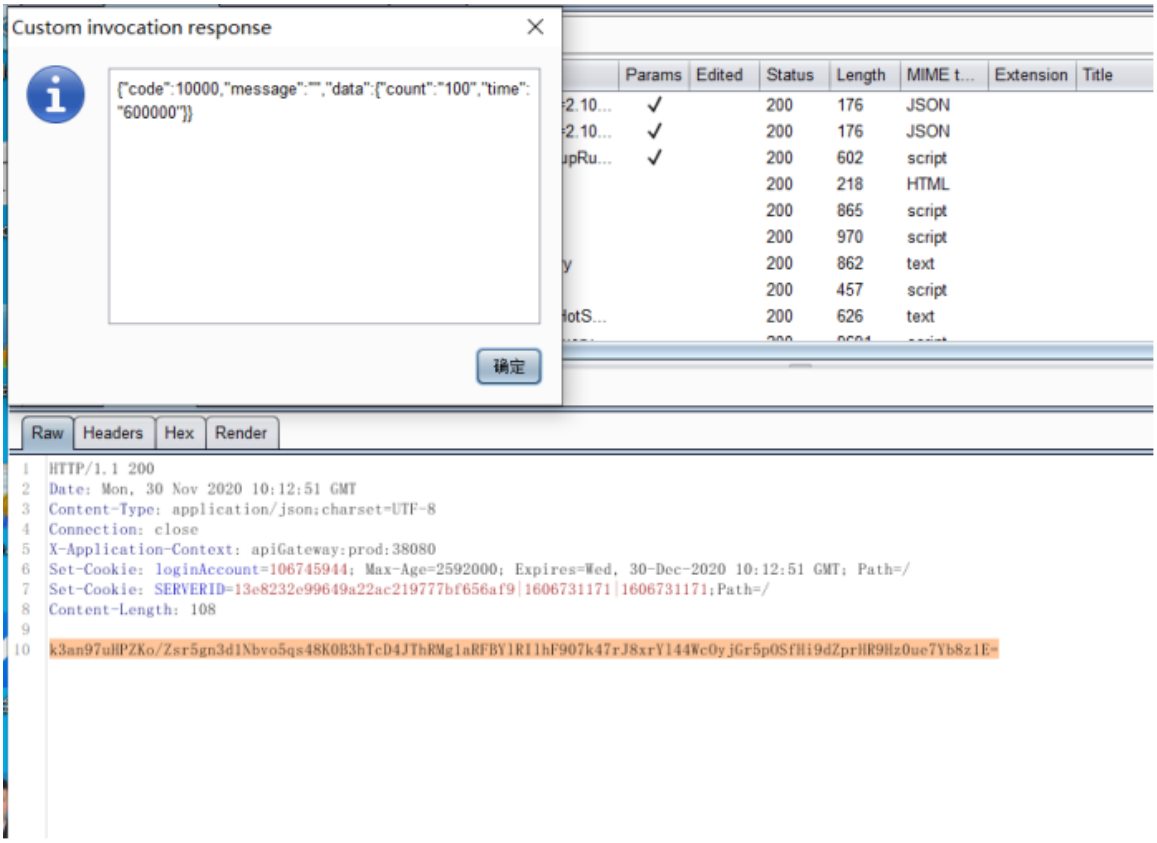
在burpsutie中进行配置。



点击send to HttpDecrypt 打开配置界面。



然后就可以对加密后的数据进行加解密了。



耗子为知，好好反思

app测试的时候，hook的难点在于定位所需要hook的函数和参数。然后就是根据函数名和参数名进行hook代码的编写。

HttpDecrypt提供了较为方便的集成的环境。相比脱壳看代码，这种调试的方法会省下我们大量的时间。

精选留言

用户设置不下载评论