# 概述

本手册主要针对微信小程序与公众号渗透测试工具、方法进行收集、整理，并结合在线资料及测试实例，旨在为测试人员提供漏洞测试的基本思路，指导微信小程序与公众号渗透测试工作。

# 工具介绍

## 反汇编/反编译工具

1. **夜神：**夜神版本andriod 7.0以下
2. **ADB**
3. **wxappUnpacker**

一款微信小程序反编译工具， 基于nodejs的反编译工具。

链接：<https://github.com/xuedingmiaojun/wxappUnpacker>

## 调试工具

1. **微信小程序开发工具**
2. **Xposed**
3. **Sqlitecipher爆破查看工具**

## 抓包工具及方案

Burp+微信。

退出微信，设置微信代理为burp监听地址。

Burp+Proxifier

使用Proxifier将微信程序的流量转发到burp

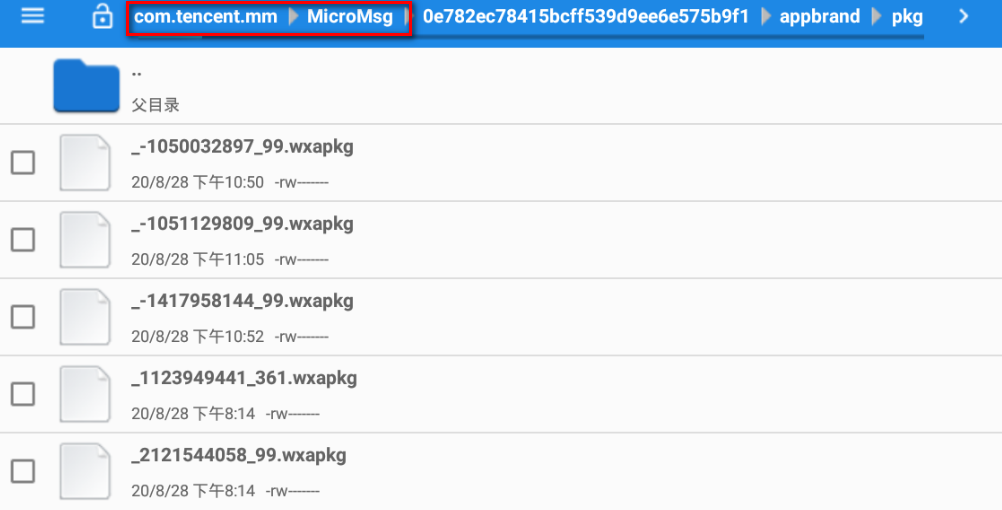
# 信息收集

## 反编译

（1）导出wxapkg文件

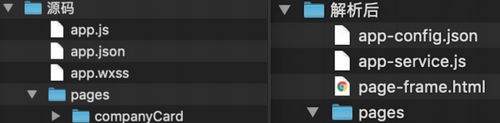
A. 通过微信运行小程序，然后点击所有小程序中的使用功能（目的是将全部wxapkg导出）

B.查看文件导出位置：/data/data/com.tencent.mm/MicroMsg/（小程序对应随机字符串目录）/appbrand/pkg



C.通过ADB导出wxapkg文件：cat \_xxxx.wxapkg > /sdcard/Download/xxxxx.wxapkg

D.文件导出完成

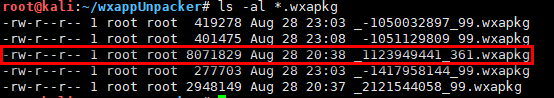


在微信服务器会将⼩程序源码中所有的“js”⽂件压⼊“app-service.js”⽂件中，将所有的“json”⽂件 压⼊“app-config.json”中，将所有的“wxml”⽂件压⼊“page-frame.html”⽂件中，“wxss”则在处理之后 以“html”⽂件的形式存留在对应⻚⾯⽬录之下。

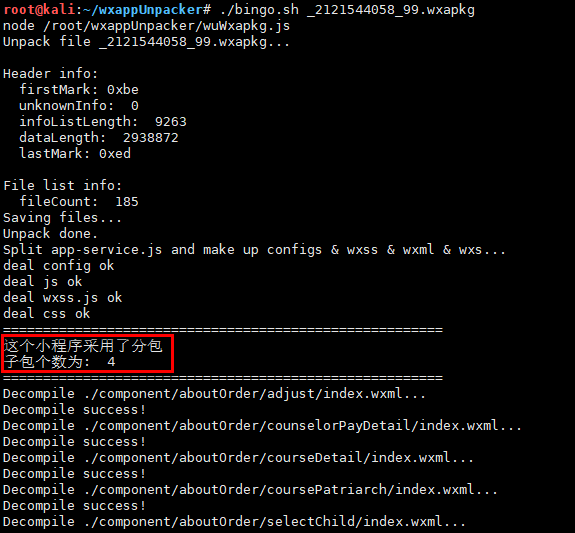
反编译要做的事情：JS数据还原、JSON数据还原、WXSS数据还原、WXML数据还原

（3）使用wxappUnpacker反编译

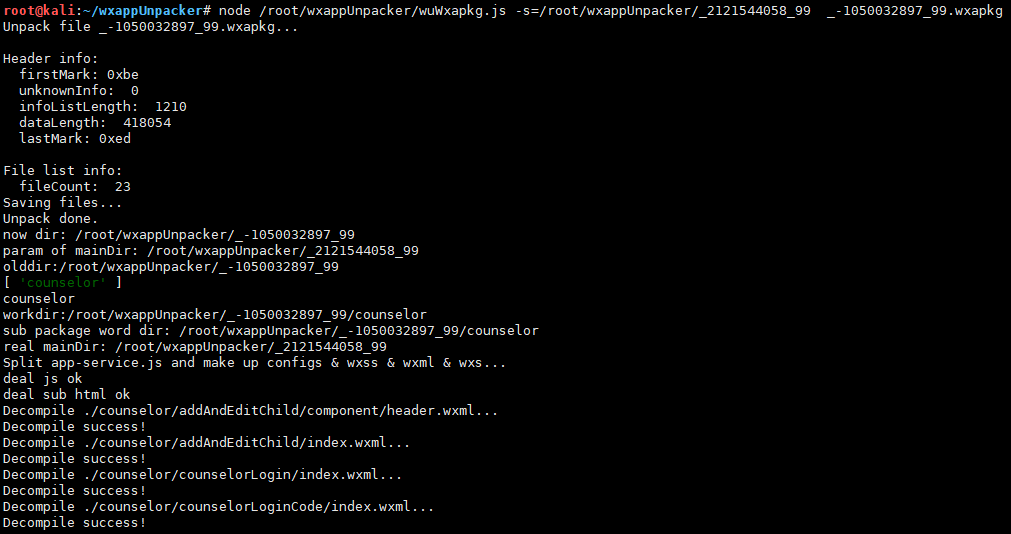
A.导出的文件：大致10M这个为微信基础包不需要反编译，\_-这个形式大致为子包，属于多个包分包情况

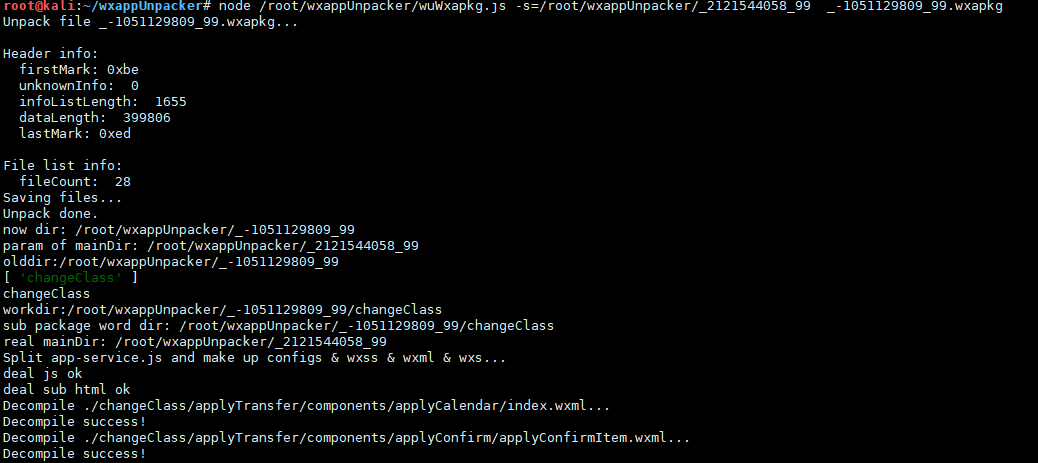


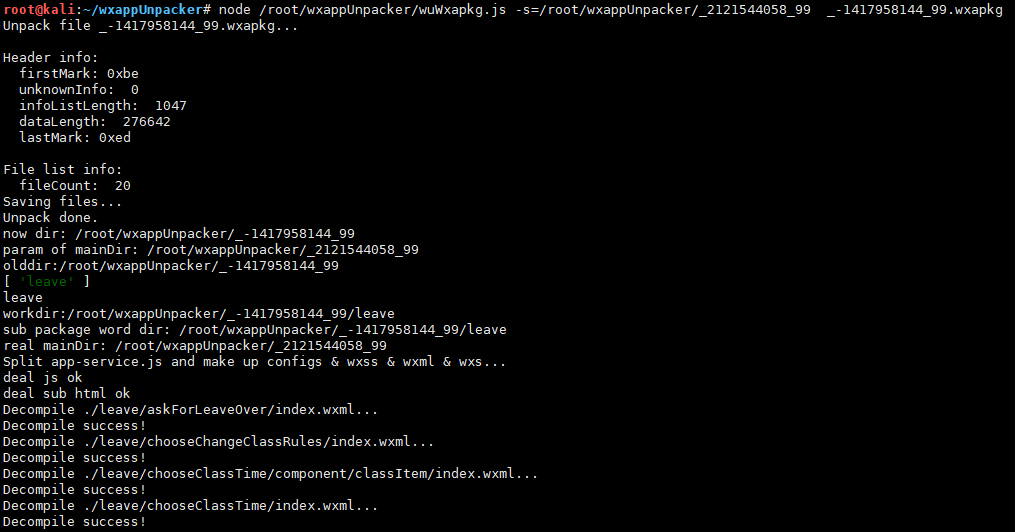
B.对于主包进行反编译，知道存在4个子包的情况（如果发现子包不全，继续操作小程序功能，看是否有导出，很多子包是需要被操作到才会导出的）



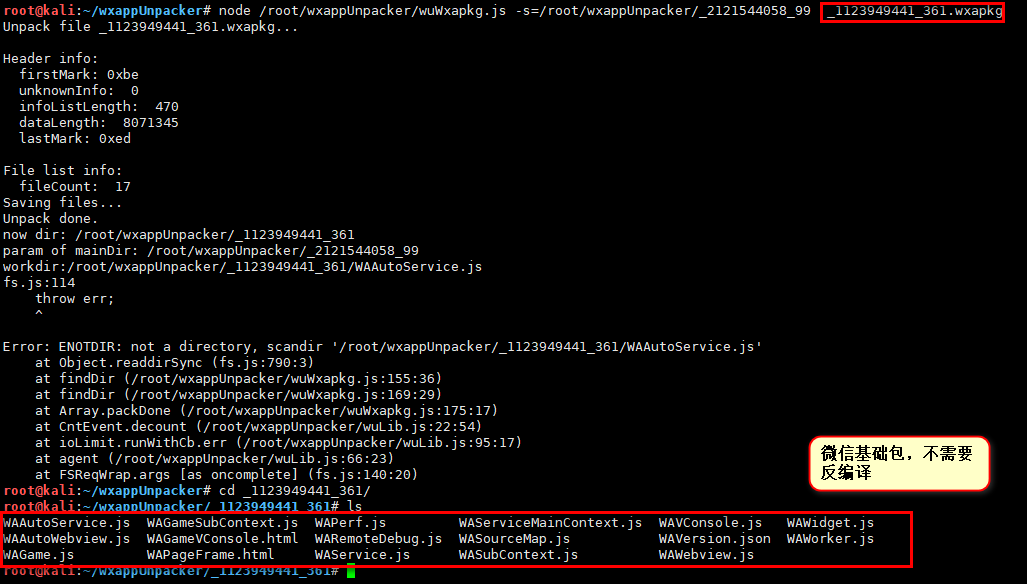
C.对子包进行反编译操作





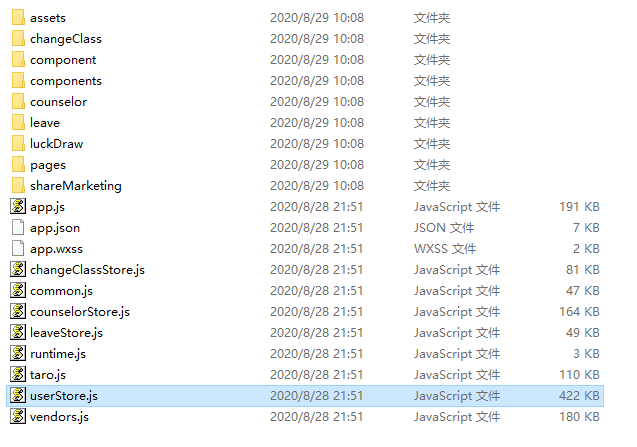


D.微信基础包不需要反编译的判断方法



E.获得反编译后的结果





## Sessionkey的判断

（1）从代码中获得的appid、secret，从抓包中获得js\_code

（2）从抓包中获得js\_code，appid、secret

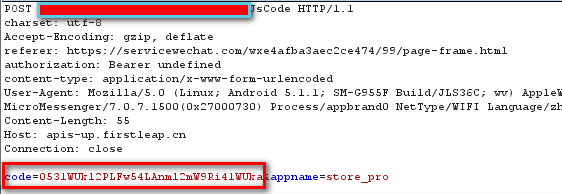
（3）从代码中获得的appid，从抓包中获得js\_code，secret

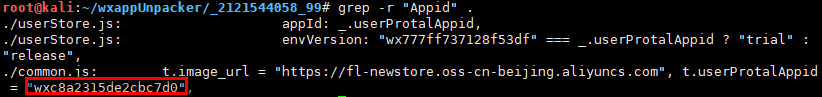
微信官⽅ jscode2session的API接口：

[https://api.weixin.qq.com/sns/jscode2session?appid={0}&secret={1}&js\_code={2}&grant\_type=authorization\_code](https://api.weixin.qq.com/sns/jscode2session?appid=%7b0%7d&secret=%7b1%7d&js_code=%7b2%7d&grant_type=authorization_code)

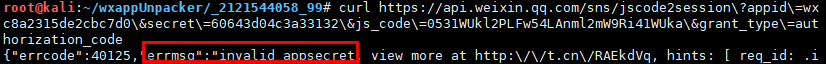
例如：验证secret是否正确:

A.抓包获得js\_code



B.代码中获得appid

C.从代码或抓包中找到secret，这里未找到，构造的secret，提示错误如图1，如果找到确认是正确返回如图2



图[1]



图[2]

## 传输内容加解密

前提：获得到session\_key

微信使⽤的是AES加密，CBC模式采⽤PKCS7填充，数据块⻓ 度为128位，输出使⽤base64编码。AES是⽤于替代DES加密的对称加密算法，对称加密算法最⼤的特 性便是加、解密使⽤的同是⼀串密钥，只要获取到key便可加解⾃如（CBC加密模式下还需要 iv 偏移量参数，⼀般固定不变或作为变量内容传输）

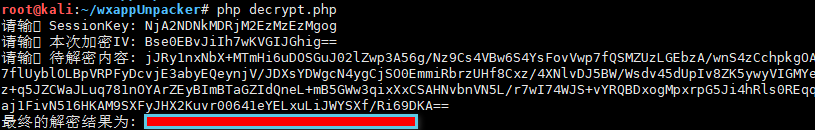
仅在“获取⽤户信息”功能中才会⽣成签名，⽤于防止内容被篡改，其他没有签名，例如：“获取⼿机号”功能中并⽆，因为签名是采⽤ sha1( rawData + sessionkey ) ⽅式⽣成的，若session\_key 泄露则⼀切验证签名的操作均无效的。

例如：解密内容

A.获得IV及加密内容，通过抓包获得



B.运行输入：session\_key、IV、加密内容



# 漏洞挖掘

## 信息泄露类漏洞

### AppSecret泄露

漏洞描述：

在⼩程序包内的配置⽂件中、博⽂、 GitHub仓库等等，找到被泄露的AppSecret。

漏洞场景：

小程序代码。

漏洞级别：

**【高危】**

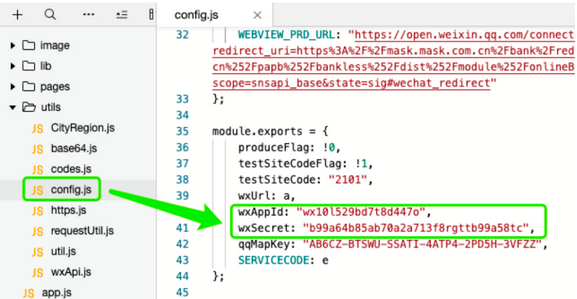
漏洞测试：

1、通过wxappUnpacker反编译工具进行分析，查看代码中是否存在AppSecret情况。

2、通过正则、关键字对代码、配置文件进行检索：检索常见的登录凭证，例如appid、secret、key等；

例如：

A. 对小程序进行反编译，再通过关键字检索，发现代码中的AppSecret。



B. 通过微信官⽅ jscode2session的API接口，判断sessionkey的方式确定此值是否为AppSecret



漏洞修复：

1. 禁止在代码中存储如口令和密钥之类的敏感数据，容易导致泄密，仅加密的密钥可以硬编码在代码中；
2. 开发者服务器可以根据⽤户标识来⽣成⾃定义登录态，⽤于后续业务逻辑中前后端交互时识别⽤户身份；
3. 会话密钥 session\_key 是对⽤户数据进⾏ 加密签名的密钥。为了应⽤⾃身的数据安全，开发者 服务器不应该把会话密钥下发到⼩程序，也不应该对外提供这个密钥。

### 请求直接返回SessionKey泄露

漏洞描述：

当开发者⾮常规范的使⽤第三⽅key来关联session\_key，并且⼀切操作查询均只使⽤第三⽅key，但也可能一些流程出现直接返回SessionKey的情况

漏洞场景：

1、在请求登录或获取openid时直接返回SessionKey，

2、在查询第三⽅key等功能中返回SessionKey

漏洞级别：

**【高危】**

漏洞测试：

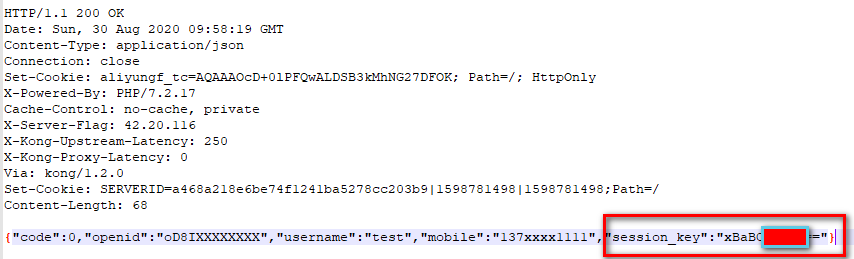
例如：

B. 在请求登录或获取openid时直接返回SessionKey

返回报文：



C. 在查询第三⽅key等功能中返回SessionKey



漏洞修复：

1）从微信这边获取到的session\_key不能直接返回给⽤户

2）如果需要返回⽤于校验⽤户身份，使用自己生成的第三⽅key，将session\_key和 自己生成的key在数据库中做个关联

3）禁止在一切查询提交等过程中通过第三⽅key返回session\_key

## Js\_code重复使用

漏洞描述：

当开发者⾮常规范的使⽤第三⽅key来关联session\_key，并且⼀切操作查询均只使⽤第三⽅key，但也可能一些流程出现直接返回SessionKey的情况

漏洞场景：

1、爆破SessionKey

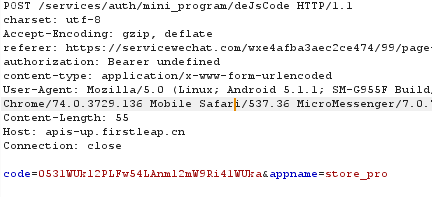
漏洞级别：

**【低危】**

漏洞测试：

例如：

A.抓包获取js\_code



B. 通过jscode2session的API接口猜解secret爆破SessionKey

[https://api.weixin.qq.com/sns/jscode2session?appid={0}&secret={1}&js\_code={2}&grant\_type=authorization\_code](https://api.weixin.qq.com/sns/jscode2session?appid=%7b0%7d&secret=%7b1%7d&js_code=%7b2%7d&grant_type=authorization_code)

返回报文：



漏洞修复：

1）每⼀次调⽤都要⽣成⼀个⼀次性有效的 js\_code 参数

2）⽣成的 js\_code 具有有效时间

3）同时间只能有⼀个 js\_code 有效（例如⼩程序内连续⽣成两 个code A和B，那么⽼的A便不再有效）

## WEB相关漏洞



### 需要关注的

描述：

各类逻辑漏洞、水平越权、SQL注入、任意文件上传、SSRF、通用组件漏洞、命令执行、反序列化

### 不需要关注的

描述：

XSS（分享朋友圈方式可以尝试）、CSRF、JSON劫持、CORS劫持、URL跳转、CRLF注⼊、目录遍历。

**原因：**

1、微信⼩程序的特性是不能执⾏动态脚本；

2、⼩程序内⽤户的⼀切操作 都是在微信沙盒内完成的，攻击者⽆法通过浏览器利⽤⽤户在⼩程序内的身份凭据来完成攻击；

3、后端API的特性（⼤多以JSON格式返回数据）