1.微信小程序安全测试指南

1. 1. 一.概述

1.1.1 指南概述

1.1.2 注意事项

1.由于新技术新业务的发展速度较快，可能存在客户要求的测试项不在指南中的情况，本文档只提供对微信小程序进行测试的基本思路；另外测试方法可能会有其他更好的替代方案，请积极探索并实践；

2.风险等级评定请按照业务需求评估，给出的定级进攻参考；

1.2. 二.测试准备

本节主要介绍微信小程序在测试过程中使用到的系统环境及测试工具：

系统
Mac OS
Windows 10

1.2.1 常用工具列表

名称	用途
iTools	下载安装 ipa，文件管理、越狱等
ADB	PC 端与安卓手机进行调试交互
BurpSuite	抓包分析工具
Nmap	端口扫描工具

名称	用途
AWVS	WEB 漏洞扫描工具
Sqlmap	SQL 注入工具
中国菜刀	Webshell 连接工具
wxappUnpacker	小程序源代码获取工具

1.2.3 微信小程序介绍

微信小程序是一种不需要下载安装即可使用的应用，它实现了应用“触手可及”的梦想，用户扫一扫或者搜一下即可打开应用，也体现了“用完即走”的理念，用户不用关心是否安装太多应用的问题。应用将无处不在，随时可用，但又无需安装卸载。对于开发者而言，小程序开发门槛相对较低，难度不及 APP，能够满足简单的基础应用，适合生活服务类线下商铺以及非刚需低频应用的转换。小程序能够实现消息通知、线下扫码、公众号关联等七大功能。其中，通过公众号关联，用户可以实现公众号与小程序之间相互跳转。 微信小程序与 APP 之间的区别

(1)下载安装

微信小程序：通过微信(扫描二维码、搜索、分享)即可获得;

App：从应用商店(App Store、应用汇等)下载安装;

(2)内存占用

微信小程序：无需安装，和微信共用内存使用，占用内存空间忽略不计;

App：安装于手机内存，一直占用内存空间，太多的 App 可能会导致内存不足;

(3)手机适配

微信小程序：一次开发，多终端适配;

App：需适配各种主流手机，开发成本大;

(4)产品发布

微信小程序：提交到微信公众平台审核，云推送;

App：向十几个应用商店提交审核，且各应用商店所需资料不一样，非常繁琐;

(5)功能区别

微信小程序：限于微信平台提供的功能;

App：对硬件资源的利用更加淋漓尽致，可以做出功能、设计、效果和流畅程度远远超过小程序的软件和服务；

(6)传输要求

微信小程序：必须使用 HTTPS，且绑定域名需要备案，不能直接使用 IP 作为地址；

App：依照开发商自主要求，HTTPS 传输可选可不选；

(7)开发背景

微信小程序：适合初创团队，试错成本低，需要较少时间和资金投入；

App：适合成熟的商业大公司，对自我品牌要求较高的企业。

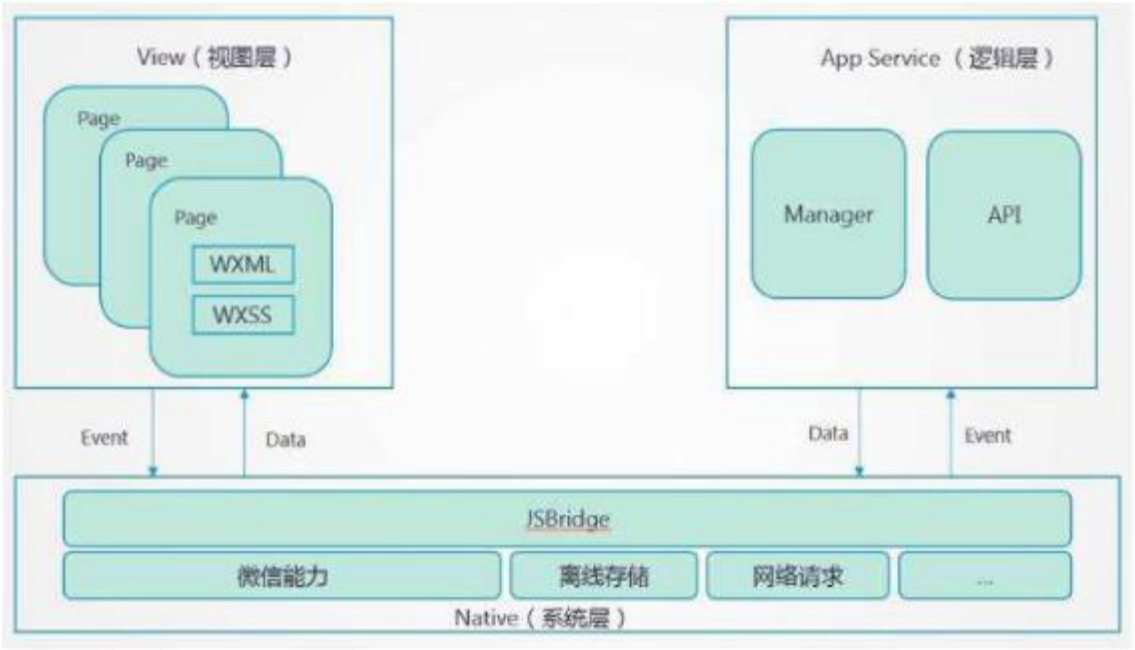
1. 3. 三.微信小程序测试指南

1.3.1 微信小程序客户端测试

3.1.1 微信小程序架构分析

微信小程序采用了框架化进行开发，主要分为视图层、逻辑层、JSBridge。JS 负责业务逻辑层的实现，而视图层则由 WXML 和 WXSS 来共同实现，前者其实就是一种微信定义的模板语言，而后者类似 CSS。

从下面的微信小程序架构图上可以清晰的看出，小程序借助的是 JSBridge 实现了对底层 API 接口的调用，所以在小程序里面开发，开发者不用太多去考虑 IOS、安卓的实现差异的问题，安心在上层的视图层和逻辑层进行开发即可



视图层

框架的视图层由 WXML 与 WXSS 编写，由组件来进行展示，将逻辑层的数据反应成视图，同时将视图层的事件发送给逻辑层。

逻辑层

逻辑层将数据进行处理后发送给视图层，同时接受视图层的事件反馈。

文件组成部分

(一)一个小程序主体部分由三个文件组成，必须放在项目的根目录：

APP.js：小程序(全局)逻辑；

APP.json：小程序(全局)公共设置，决定页面文件的路径、窗口表现、设置网络超时时间等；

APP.wxss：小程序公共(全局)样式表。

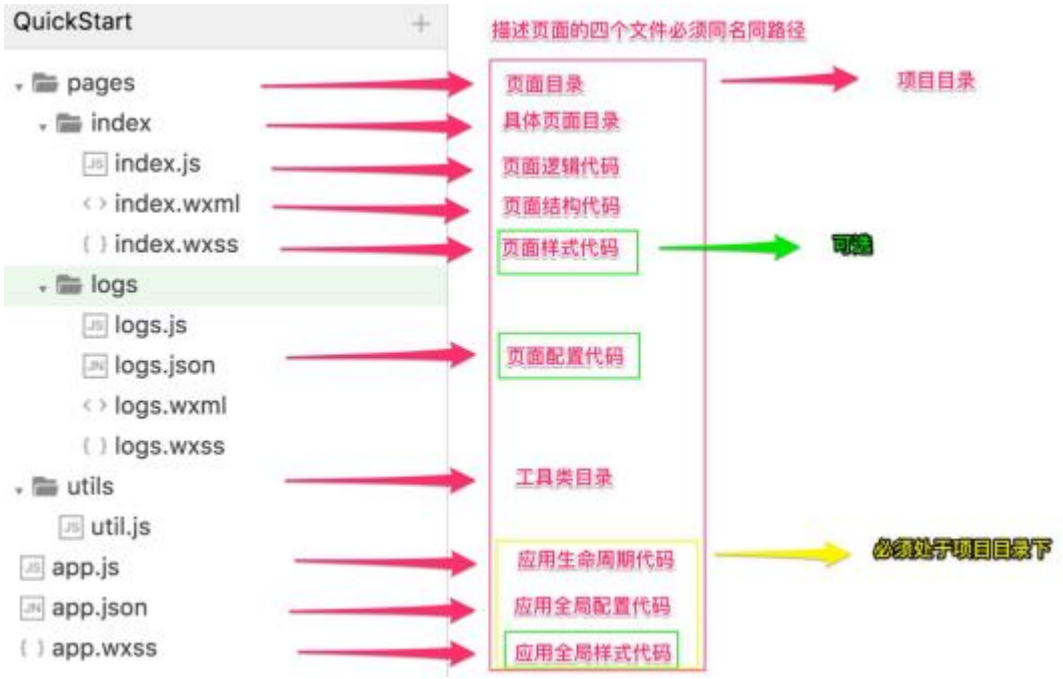
(二)一个小程序页面由四个文件组成：

JS：页面逻辑；

WXML：页面结构，框架设计的一套标签语言，结合基础组件、事件系统，可以构建出页面的结构；

WXSS：是一套样式语言，用于描述 WXML 的组件样式，用来决定 WXML 的组件应该怎么显示；

JSON：页面配置。



从上述的架构图、文件组成部分来看，重点分析的就是小程序的逻辑层。而逻辑层主要的组成部分是由 APP.js、APP.json、JS 文件、JSON 配置文件等组成，因此测试过程中主要分析的对象就是这一些。

3.1.2 微信小程序客户端功能模块安全

微信小程序客户端功能模块安全大致可以分为以下 6 个部分：

3.1.2.1 网络传输安全

微信小成熟传输虽然使用 HTTPS，并对访问域名进行校验控制，但如果后端服务器未做 SSL 双向认证，仍无法抵御攻击者在本地安装代理证书实施中间人攻击的威胁。

3.1.2.2 数据存储安全

本地数据存储采用（KEY，VALUE）形式存放在 DB，数据的保护继承了微信的数据库加密防护策略。

3.1.2.3 文件存储安全

本地文件存储采用 HASH 映射机制进行文件定位，文件存储在外部存储，本身通过自定义算法实现完整性校验。

3.1.2.4 框架本身安全：

框架上继承了微信成熟的 JSAPI 框架和底层的 TBS 浏览器内核，因此在未出现新的 0DAY 漏洞之前，整体的框架安全还是十分可靠；

3.1.2.5 伪造小程序二维码安全：

扫码功能依赖微信 APP 的原生的扫码功能；生成小程序特定页面的直达二维码，依赖于 ACCESS_TOKEN，而 ACCESS_TOKEN 是通过小程序私有的 APPID 和 APPsecret 请求得到，攻击者无法获知到 APPsecret 信息伪造生成二维码；

3.1.2.6 数据泄露安全

小程序登录体系可以依赖微信接口和公众号平台，也可以由小程序自行实现。前者根据微信平台的安全规范实施，由微信进行整体的安全维护，因此安全性较高。后者则由小程序自行控制安全性。

从上述的客户端功能模块安全分析中来看，小程序客户端本身的安全继承了微信 APP 整体的安全建设。因此无需像传统的移动端测试对客户端本身进行测试，测试的重点还是在于服务端，即小程序与后端进行交互过程中存在的安全风险，基本上跟测 WEB 没有区别。但是跟传统黑盒测试 WEB 的区别是，在目前为止，可以非常简单的提取到小程序的源码信息。如小程序在传输过程中进行了数据的加密传输，那么我们就可以通过源码的分析来得出整个加密算法的技术流程。

3.1.3 微信小程序源代码提取

微信小程序分发后的文件后缀为 pkg。在写这篇指南的时间为止，提取出来的 pkg 文件还是可以通过工具直接还原成可读的源代码，但不排除后续微信更新，增加 pkg 文件提取、逆向还原的难度。

微信小程序在手机中存放的目录如下：

安卓： /data/data/com.tencent.mm/MicroMsg/不同微信号的值不同 /appbrand/pkg

苹果： /var/mobile/Containers/Data/Application/不同微信号的值不同 /WechatPrivate/c15d9cced65acecd30d2d6522df2f973/WeApp/LocalCache/release

如果手机上打开了很多微信小程序，那么目录中就会存在多个小程序 pkg 文件。第一种区分方法是按照修改时间来进行区分。第二种方法是在微信页面中删除所有浏览过的小程序，重新打开需要进行测试的小程序，那么目录中只会存在一个 pkg 文件。

安卓的提取方法如下，苹果可以使用 Itools 工具直接拷贝。

```
> adb shell
# cd /data/data/com.tencent.mm/MicroMsg/70aa3417825137674379747 2a68c1c6a/appbrand/pkg
# ls
_-1261323258_6.wxapkg
_1079392110_3.wxapkg
_1123949441_106.wxapkg
# cp _-1261323258_6.wxapkg /sdcard/
# exit
> adb pull /sdcard/_-1261323258_6.wxapkg .
```

https://github.com/qwerty472123/wxappUnpacker 小程序逆向还原工具，依赖 nodejs 等，具体安装、使用说明可以参照链接中的内容。

常用还原源码命令：node wuWxapkg.js –o –d 目标文件.pkg

还原后的文件形式大致如下：

名称	修改日期	大小	种类
app-config.json	今天 下午7:27	2 KB	JSON
app-service.js	今天 下午7:27	91 KB	JavaScrip
app.js	今天 下午7:27	2 KB	JavaScrip
app.json	今天 下午7:27	1 KB	JSON
app.wxss	今天 下午7:27	23 KB	文稿
fmsdk	今天 下午7:27	--	文件夹
images	今天 下午7:27	--	文件夹
page-frame.html	今天 下午7:27	157 KB	HTML...oc
pages	今天 下午7:27	--	文件夹
utils	今天 下午7:27	--	文件夹
weps	今天 下午7:27	--	文件夹

文件夹中的 HTML、WXSS 等文件主要是存放了页面结构、小程序页面样式等内容，重点分析的还是文件中 JS 文件，如小程序在与后端服务器进行了加密传输，可以根据传输中的加密参数值进行跟踪，逐步分析参数是进行如何加密方式。

3.1.4 微信小程序源代码分析样例

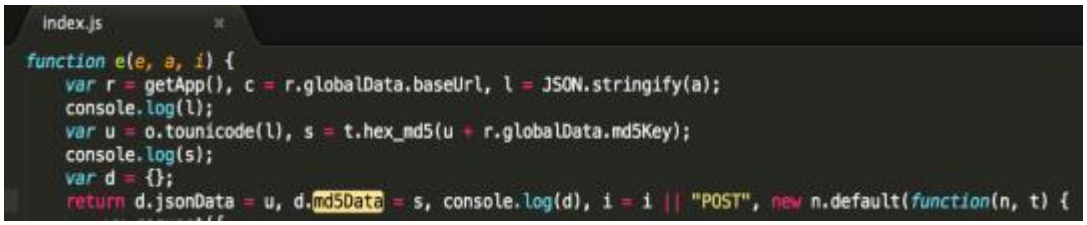
如下图中的请求包，在请求包内存在参数 md5Data。该处请求包为登录时的请求包，尝试修改手机号为其他手机号进行爆破，发现当修改了手机号之后，服务端未报错，提示验证失败。那么这个 md5Data 参数就是传输过程中的校验参数值。

如果我们能通过源代码中获取到 md5Data 参数的生成方式，那么我们就可以掌握整个传输过程中的加密方式，进而可以更加深入的对该程序进行分析。

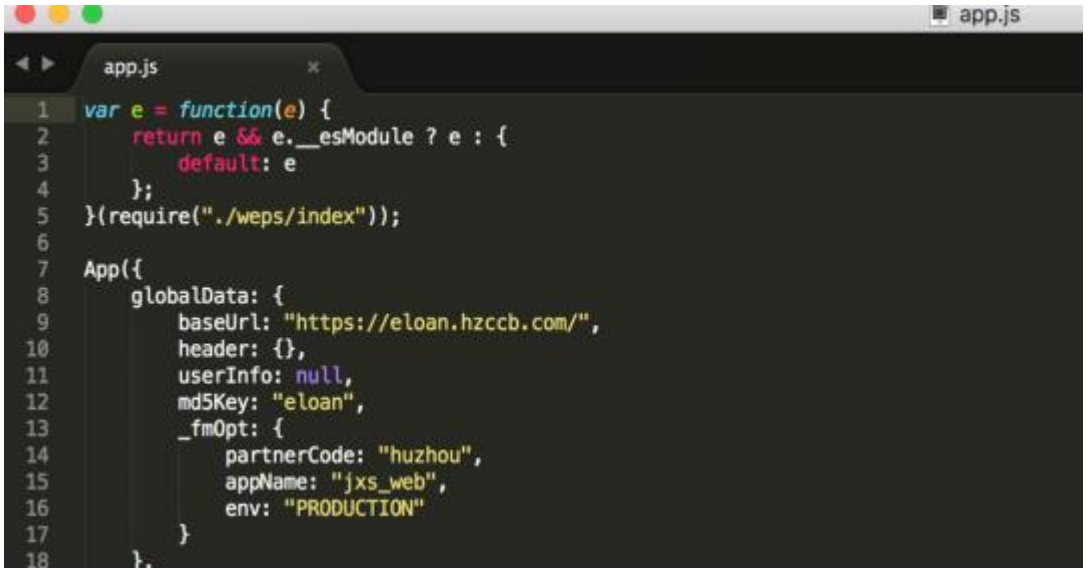
```
POST /get_sms_code HTTP/1.1
charset: utf-8
referer: https://servicewechat.com/wx49f7c11675d7c919/1/page-frame.html
content-type: application/json
User-Agent: Mozilla/5.0 (Linux; Android 8.0; MI 6 Build/OPR1.170623.027; wv)
AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/53.0.2785.143
Crosswalk/24.53.595.0 XWEB/151 MMWEBSDK/19 Mobile Safari/537.36
MicroMessenger/6.6.6.1300(0x26060637) NetType/WIFI Language/zh_CN
MicroMessenger/6.6.6.1300(0x26060637) NetType/WIFI Language/zh_CN
Content-Length: 115
Host: eloan.hzccb.com
Connection: close

{"jsonData":"{\"cellphone\":\"18042045868\",\"password\":\"123321\"}","md5Data":"f51c9
b0ef71b6809b0b199fc238d484d"}
```

首先全局搜索 md5Data 关键字，发现在 index.js 中写明了 md5Data 参数的生成过程。md5Data 参数是由对象 u+md5key 这两个参数进行生成的。



再全局搜索 md5key 这个关键字，发现中 app.js 中直接明文写出了值为 eloan



在源代码中再进行分析，发现对象 u 的值为请求包中请求对象的 json 字符串，知道了加密对象和加密值，而且整个加密过程为简单的 md5 加密，那么我们就进行确认这个加密的思路是否正确。

```
{"cellphone":"18042045868","password":"123321"}eloan

f51c9b0ef71b6809b0b199fc238d484d
```

可以看到最终生成的 md5 值与请求包中的 md5Data 的值一致，这就说明整个加密过程的分析是正确的。知道了整个加密过程，整个小程序对我们来说就是形同没有做任何加密的。因此我们可以其他漏洞的测试，如尝试进行越权查询、SQL 注入、账号密码的爆破等漏洞的测试。

整个小程序的源代码分析基本上是这样的思路，通过加密参数为关键字进行全局搜索，从而获取到整个加密方式，最终对加密方式进行比对，是否是正确的。不过有些小程序没有对传输过程进行加密，那么测试的过程就可以省去这一步。