每次不重样，带你收获最新测试技术！

微信扫一扫关注我们

扫码填问卷，免费领资料

投稿邮箱：editor@51testing.com

ADAS 测试面试题汇总

◆ 作者：狼图腾

专有名词

VRU：VulnerableRoadUsers，弱势道路使用者，包括行人、两轮车。

RT-Range：真值，用于测量实时的位置、距离、速度、方向等信息，如测试中需要用到两车之间的相对距离、速度可以在自车和目标车上装 RT-Range。

LaunchPad：LunchPad 是一个可滑动的底板，里面有机械结构、滑轮等，上面可安装假人假车，人为控制速度拖动假人假车往前走，同时可以获取其实时的距离、速度、方位角等信息。

项目及团队

1.项目背景/项目介绍

目前在做的是 xx 车辆的 ADAS 项目，项目主要分两个阶段，V1 阶段主要是是传感器、执行器的测试(工作内容：编写测试用例、评审、发布，然后根据测试用例编写测试方案，包括车辆改制，人员安排，时间安排，项目预算，设备租赁等)，之后就是根据测试用例及方案去试验场测试，测试完成进行结果分析以及测试报告的编写。V2 阶段则为 ADAS 相关功能的验证测试(AEB,ACC,LKA,LCA,BSD,DOW 等)，这个阶段也是要编写测试用例及发布。V2 阶段根据开发成熟度细分为 V2.1,V2.2,V2.3，因成熟度不同，每个阶段会选取部分测试用例安排测试计划及测试执行、测试报告编写。

2.项目人员构成及职责划分

例：①产品（摄像头、毫米波雷达、激光雷达）

②开发（系统开发/应用软件开发/底层软件开发/域控制器开发/硬件开发）

③测试（测试用例、整车测试、试验管理）30 人左右

测试用例组：对接需求，写测试用例

整车测试：实车测试，测试用例执行

试验管理：设备/车辆管理，场地预约

④产品运营中心

3.测试日常工作包含内容

基础：

1） 根据功能描述和需求完成传感器性能测试用例的编制及发布，包括 Camera，Radar，Lidar 等；

2） 完成测试方案的编制，包括车辆改制，测试计划，人员安排等；

3） 根据测试用例完成传感器性能测试；

4） 使用 CAN 工具完成传感器性能测试结果分析；

5） 编写传感器性能测试报告及评审；

6） 依据测试用例及需求说明书完成 AEB/ACC/LKA 等功能测试规范的编制及发布；

7） 根据开发进度完成各功能测试计划安排；

8） 根据测试规范完成功能验证测试；

9） 完成功能测试结果分析并跟进问题改进进度；

10） 测试报告的编制、评审；

11） 负责自动驾驶系统道路测试工作，记录道路测试问题，形成道路测试报告；

12） 分析道路测试问题，对测试问题进行分类、管理和跟踪；

13） 协助研发团队进行研发测试和问题复现；

14） 完成上级领导安排的其他任务，如：VRU 驱动系统使用说明书、Lidar 操作指导书的编制、传感器标定测试等。

进阶：

1） 负责团队测试工具框架搭建，编写测试数据生成、测试日志分析等测试辅助工具；

2） 负责建立自动化测试框架，协助完成自动化脚本及用例的编写；

3） MATLAB/simulink/carsim 完成无人车运动学和动力学的建模；

4） 利用 carsim 和 simulink 联调实现车辆队列模型通信及构型；

5） 使用 PID/LQR 等控制控制算法最终实现车道保持辅助功能。

4.项目周期

例：ADAS 从立项到交付正常要一年半到 2 年多。

5.测试报告内容

①试验目的、试验规范；

②试验设备(包括被测车辆信息，数采系统，目标物，真值系统等)；

例：车辆型号、传感器版本、执行器版本、域控制器版本、试验前后车辆里程

③试验时间、地点、人员；

④试验环境(天气，光照，路面等信息)；

⑤试验结果(以表格形式列出每条用例实际测试的结果分析)。

传感器性能测试

1.测试流程

例：首先对接需求，根据需求完成相关传感器性能测试用例的编写及评审，然后编写测试方案，包括测试计划及车辆改制等，测试计划主要是根据测试用例安排测试所需要的时间计划，车辆改制则是根据需求把传感器安装到试验车指定位置，前置毫米波雷达一般是安装在前保险杠上，激光雷达一般安装在车顶，然后接网关之类的，需要读取到车辆底盘信号,车身信号以及雷达信号,到试验场还要租赁安装真值系统,及 launchpad、假人假车等。剩下的就是按照测试用例执行。（V2 阶段的各功能测试则不需要再安装传感器等，此阶段传感器/执行器啥均已经集成到试验车）

2.车辆改制实车环境搭建

事项：车辆准备、传感器安装规范、传感器支架制作、传感器线束、电源（12V/24V/220V）、backbone（车身/主干网）、ChassisCAN（底盘）网关转发车辆信号到雷达。

3.哪些目标物可以装 RT-Range

只有目标车装真值，行人/两轮车使用的是假人/假自行车装在 LaunchPAD 上，可以获取距离/速度/方位角等信息。

4.传感器性能怎么测？

在车上接上真值、雷达、底盘等线束，保证可以读取到这些数据，然后在测试机的 CAN 工具中录制数据，按照测试用例的步骤执行，所有步骤完成了，结束录制，保存数据即可。

5.录制的软件

Radar：CANOE 工具，雷达配的上位机软件(如果有的话)，以及记录场景的摄像头。

(类似于行车记录仪)

Lidar：上位机软件用于显示实时点云图及回放功能；采集用 Wireshark。

6.传感器性能测试有哪些场景？

毫米波雷达	①FOV（包括横穿/纵穿，行人/两轮车/目标车） ②分辨率（距离分辨率、角度分辨率、速度分辨率） ③动态跟车（接近、远离、直道、弯道、cutIn、cutOut） ④最远探测距离（行人、两轮车、目标车）
激光雷达	城市、郊野、高速道路、阳光直射、交通标志牌、轮胎、锥桶
摄像头	车道线识别、目标物类型识别等，其他测试场景同毫米波雷达

7.各传感器安装位置，有效距离

毫米波雷达	①FOV（包括横穿/纵穿，行人/两轮车/目标车） ②分辨率（距离分辨率、角度分辨率、速度分辨率） ③动态跟车（接近、远离、直道、弯道、cutIn、cutOut） ④最远探测距离（行人、两轮车、目标车）
激光雷达	城市、郊野、高速道路、阳光直射、交通标志牌、轮胎、锥桶
摄像头	车道线识别、目标物类型识别等，其他测试场景同毫米波雷达

8.各传感器优缺点

	摄像头 Camrea	激光雷达 Lidar	毫米波雷达 Radar	超声波传感器
成本	☆	☆☆☆	☆☆	☆
探测距离	☆☆	☆☆	☆☆☆	☆
测距功能	☆	☆☆☆	☆☆☆	☆☆
测速功能	☆	☆	☆☆☆	☆☆
分辨率/精度	☆☆☆	☆☆	☆☆☆	☆☆
夜视效果	☆	☆☆☆	☆☆☆	☆☆☆
恶劣天气	☆	☆	☆☆☆	☆
行人判别	☆☆☆	☆☆☆	×	☆☆
路标识别	√	×	×	×
优势	物体分类 车道线检测	3D 感知	全天候	侧方超车/倒车入库

9.传感器是否需要标定？

测试前是否需要标定，还是要看用的哪家供应商的传感器，有的需要自己标定，有的供应商已经标定好。

例：xx 供应商的雷达的标定使用的是自己的标定软件，过程也比较简单，在车辆驾驶状态下发送一串 16 进制的数组就行，同样可通过发送 16 进制的数组可以读取到标定是否 100%。摄像头的标定有内参标定和外参标定，标定的目的主要有两个，一个是消除畸变，一个是统一坐标系，标定就是为了获取内/外参数矩阵，用的是张氏标定法，通过拍摄 N 组棋盘格的照片实现。此外还有其他的标定方法，比如基于自然场景的标定。

10.坐标系是如何确定的？坐标原点一般在哪？

在 ADAS 测试中一般会统一采用右手坐标系，x 轴朝前，y 轴朝左，z 轴朝上。常用的坐标原点有前/后轴中点（真值系统等）、前保险杠中心（传感器）。

11.激光雷达关注那些问题？

①交通标志牌等高反物体的点云质量是否出现上下延伸、前后拖点现象；

②物体边缘是否会出现不明点云；

③高速运动中观察周围路灯杆是否倾斜、卷曲、断裂等情况；

④阳光直射场景噪点是否会增多。

12.数据如何传输（传输方式）

摄像头：①视频信号转换为 CAN 信号直接输出。

②通过千兆网卡实现通信。

毫米波雷达：通过 CAN 总线给下游发送消息。（数据格式为 blf，binarylogfile,一种二进制文件）

激光雷达：以太网。（CAN 传输速度最大只有 1M/s,激光雷达点云数据量大，因此用以太网）

CAN 总线最大传输速率是 1M/s,以太网 10M/s，毫米波雷达采集的数据不是很大，采一段数据大概几十兆，所以 CAN 够用，但激光雷达的点云数据非常大，录一两分钟的数据就有几百兆数据，所以用以太网；LIN 速度只有几十 kb/s，一般用在：门，方向盘，座椅，空调，照明等。

13.数据分析,分析哪些方面？报文看那些？如何分析？

数据分析主要看传感器采集的数据和真值的误差 Dx/Dy/Vx/Vy(横纵向距离和横纵向速度)，计算平均值、方差。

分析的过程：首先在 CANOE 工具里导入 dbc 文件，包括车身 dbc，底盘 dbc，真值 dbc 等(dbc 就是 CAN 数据库，里面定义了节点、报文及对应信号的定义及说明等信息)，然后将测试采集到的数据(blf 格式)通过工具转成 csv 格式导出，之后在 csv 文件中筛选出真值和雷达相关数据，主要是横纵向距离及速度，进行比较，计算平均值和方差。

14.数据分析，误差多少是合格的

例：距离误差在 0.7m 之内。

15.测试对天气的要求

天气一般是在晴朗的天气下进行的，测试规范里有对天气的要求：光照强度不低于 20000lux，夜间实验光照强度不高于 11lux，水平能见度大于 1km。

可参考 C-NCAP2021；此外由于不同传感器在恶劣环境下的表现不同，也会适当涉及雨雾冰雪这种天气。

16.回归测试策略

根据性能测试结果分析哪些场景下传感器的性能表现较差，挑选典型测试用例进行回归测试。

17.传感器性能如何评价的？（评价术语）

传感器性能测试：横纵向位置测量准确、有效 FOV 略小、原距离下 FOV 表现不理想，目标物初次稳定识别的距离散布很大、测试数据没有明显规律、中远距离出现目标短暂丢失现象、对不同目标物的最远探测距离都很稳定、左右两侧误差分布更为集中、横向误差一般且随纵向距离增大而迅速增大不利于目标物的融合、在各个距离段误差一致性更好同时测量的精密度更高，有利于通过补偿缩小系统误差……

功能测试

1.ADAS 工作原理（功能逻辑/技术要点）

首先依赖感知传感器对道路周边环境信息进行采集，包括摄像头、激光雷达、毫米波雷达和超声波传感器、高精地图、GNSS 卫星定位、IMU 惯性导航等。采集的数据传输到中央计算单元进行计算，用来识别自车周边障碍物的状态和可行驶区域，制定相应控制策略，替代人类做出驾驶决策，(如路径规划等)。最后是执行控制模块制定方向盘转角、线控加减速、线控制动等信息，传输到底盘执行机构，按照指令进行精确执行。

2.各功能都需要用到哪些传感器

多数功能都是依靠的前置传感器(经过 Lidar、Radar、Camera 融合)，比如 ACC,AEB,LKA,NOA,TJA,FCW 等，只有少数涉及周视传感器，比如 BSD,CVW,DOW。

级别	实现功能	典型配置	常规主流配置方案
L4	AVP代客泊车	5V12U	1前向4环视+4低频超声波 8高频超声波
L2+/L3	C Pilot部分城市场景领航	9V5R	3目前向4侧视1后视1监测+1前向4角雷达
	TJP交通拥堵领航	3V5R	3目前向摄像头+1前向4角雷达
	HWP高速公路领航	3V5R	3目前向摄像头+1前向4角雷达
	VPA记忆泊车	4V12U	4环视+4低频超声波 8高频超声波
L2	HWA高速公路辅助	1V5R	1前向摄像头+1前向毫米波4角雷达
	ALC自动变道辅助	1V5R	1前向摄像头+5前向毫米波
	ICC智能自适用巡航	1V1R	1前向摄像头+1前向毫米波
	ICA集成式巡航辅助	1V1R	1前向摄像头+1前向毫米波
	TJA交通拥堵辅助	1V1R	1前向摄像头+1前向毫米波
	RPA远程泊车辅助	12U	4低频超声波 8高频超声波+车载蓝牙
	APA自动泊车辅助	12U	4低频超声波 8高频超声波
L1	ACC自适应巡航	1V/1R	1前向摄像头/1前向毫米波
	LKA车道保持辅助	1V	1前向摄像头
	AEB自动刹车	1V/1R	1前向摄像头/1前向毫米波
L0	LDW车道偏离预警	1V	1前向摄像头
	CC定速巡航	无	无

3.用例规模

全称	简称	测试用例数
自适应巡航	ACC	50+
自动紧急制动	AEB	约 30
前向碰撞预警	FCW	<10
紧急转向辅助	EMA	<10
后碰撞预警	CMSR	<10
车道保持	LKA	约 10
车道偏离预警	LDW	<10
高速辅助	HWA	约 20
领航辅助	NOA	20
盲区监测	BSD	30+
变道辅助	LCA	30+
交通标志识别	TSI	约 30
开门预警	DOW	约 40
交通信号灯提醒	TLA	20+
前方交叉交通报警	FCTA	约 20
后方交叉交通报警	RCTA	约 20

4.ADAS 各功能测试顺序、依据

说明：功能测试一般先测与前视传感器相关功能，最后测与环视/测试传感器相关功能。

①ACC、AEB、LKA。（跟前视传感器相关）

②BSD、LKA、DOW。（跟侧视/环视传感器相关）

5.一个功能验证需要多长时间？

例：与测试用例规模有关，少则四五天，多则十几天。

6.一天执行多少条用例？

例：8-10 条/天

（测试前需要调试设备及测试过程中可能会遇到设备故障等问题，因此一天执行的测试用例数不会太多）

7.测试过程中发现什么问题？痛点是什么？

数据只采集一次可能会出现无效数据，所以通常一条用例执行两次。

8.测试用例有多少？

按功能分，少则七八条，多则四五十条。

9.功能验证迭代周期？迭代哪些内容？

周期：少则半个月，多则一个月。

迭代内容：底层通讯、网络管理、诊断、刷写、信息安全、OTA。

10.ACC 原理？预期结果？

原理：设定跟车距离和速度跟随前车行驶。

预期结果有：

①设定车速和跟车距离后会有加速到指定车速的时间范围，通常是 20 几秒。

②达到设定速度后变化率要小于 1km 之类的，跟车距离同理。

③如果前面有车切速度小于自车车速，自车减速对减速度有要求，比如 3.5m/m2。

④如果前车静止了，自车有个跟停距离要求，比如 2-3m。

11.ACC 弯道怎么跟车？

弧度过大,ACC 不会生效。

试验场有不同曲率的弯道，比如 125m,250m,500m 的，根据需求跟车行驶，看是否会减速，能不能驶出弯道等。

12.ACC 采集信号有哪些？

距离和速度（横纵向）、加速度、航向角等。

13.ACC 跟车，假如到红绿灯，前方车辆突然刹车，自车却没有刹车，这是什么原因导致的呢？

距离过近，导致超出 ACC 生效范围。（ADAS 各功能生效都有严格的条件）

14.AEB 制动安全距离？

例：需求文档中要全为 0.5m——2m。

15.AEB 目标物选取原则？

无论是目标车还是行人、两轮车，都会选用假目标物代替。（真人/真车易发生事故）

16.AEB 测试场景？

①车辆追尾自动紧急制动系统（AEBCCR）

AEBCCR 系统包含两种测试场景：CCRs、CCRm：

测试场景	测试速度（km/h）	偏置率
CCRs（前车静止）	20	-50%
	20	100%
	30	+50%
	30	100%
	40	-50%
	40	100%
CCRm（前车慢行）	30	+50%
	30	100%
	40	-50%
	40	100%
	50	+50%
	50	100%

②行人自动紧急制动系统（AEBVRU_Ped）

AEB VRU_Ped	CPFA-25	CPFA-50	CPNA-25	CPNA-75	CPLA-50
测试速度（km/h）	20-60				
目标物速度（km/h）	6.5		5		
光照条件	白天和夜晚	白天			白天和夜晚
车灯状况	近光灯	/			远光灯
路灯状况	路灯				无路灯

③二轮车自动紧急制动系统（AEBVRU_TW）

AEB 两轮车	CBNA	CSFA	CBLA
VUT 速度（km/h）	20-60	30-60	20-60
目标物速度（km/h）	15	20	15
碰撞位置	50%	50%	50%
光照条件	白天	白天	白天

17.AEB 鬼探头，前方突然出现行人，触发 AEB 却没刹住，这是什么原因呢？

①测试场景超出 AEB 触发范围。

②传感器性能问题，如 Radar 本身对行人，尤其是静止物体识别能力较差。

18.EMA 测试场景有哪些？

例：前车速度 10kph，自车速度 60kph，两车相距 150m,偏置率 50%或 100%，FCW 功能触发后 1s 内以 150°/s 的速度转动方向盘，转动角度为 15°。

其他场景还有前车横穿以及前车为两轮车；其他参数保持不变。

19.EBA 和 AEB 的区别？

EBA 可以看成 AEB 的加强版，当 AEB 自动紧急制动时制动力不足时 EBA 会介入增加制动力。

20.LDW 测试场景？预期结果？

测试场景：包括直道和弯道，还有实线和虚线。

预期结果：报警时刻不晚于车轮外沿到车道线 0.2m。

21.法规有哪些？

- C-NCAP
- E-NCAP
- I-Vista

22.测试中出现 BUG 问题如何处理？

JIRA 中建问题单。

23.问题单如何建？有哪些要素？

例：问题单主要字段描述如下：

字段	说明/描述	
Project	项目标题可以表述为：[车企名称]_[项目代号]	
Issue Type	Bug	
Issue	测试用例名称	
Summary	[功能名称][测试方法][车辆 ID]Bug 描述] 例如：[ACC][整车测试][#101]ACC 无法触发跟车行驶	
Components	该 Bug 属于哪个模块，如：应用软件/底层软件/硬件/产品等	
Labels	格式：功能名称，测试类型，测试方法 功能包含：ACC/AEB/FCW/LKA/NOA/BSD/LCA/DOW 等 测试类型包括：SmokeTest/FunctionTest/PerformanceTest/RegulationTest 等 测试方法主要有：实车测试/HIL 台架测试	
Description	1. 天气，测试地点 2. 车辆 ID，车辆状态 3. 测试用例描述 4. 预期结果 5. 实际结果 6. 日志 7. dbc 名称	例：1. 晴朗，上海公共道路 2. #101，空载 3. ACC 设定 30kph 的速度跟随前方 20kph 速度的目标车 4. 自车以 20kph 速度跟随前车行驶 5. 自车未减速，发生碰撞风险 6. 数据链接路径 7. 使用 Chassis、BackBone、Safety 的 dbc 文件名

		称
Severity	1. Block：阻塞，无法进行后续试验 2. Critical：部分功能缺失，完全无性能 3. Major：功能正常，性能极差 4. Minor：功能正常，性能无法完全满足客户需求	
Environment	1. 车辆 ID ，测试阶段 2. ADCU(域控)软硬件版本（SW&HW） 3. 视觉传感器版本 4. EPS/ESC 版本	例： 1. #101，V1.0 阶段 2. ADCU HW: V1.0, SW V1.0 3. SW：Vxxx(摄像头/雷达版本号) 4. NA
Linked Issues	该 Bug 由谁确认，一般为上级领导	
Assignee	解决该 Bug 需要分配给的下一级处理人	

24.波特率的作用？设置多少？

波特率越高，通讯距离有限（制约距离的原因是信号延迟）。

两节点之间波特率偏差较大会导致通讯失败，我们测试时车身、底盘、真值都设置为 500。

拓展学习

[1] 自动驾驶测试用例数量太大怎么办？

https://www.atstudy.com/course/1012021

[2] 整车测试/座舱域测试/ADAS 测试系统学习

咨询：400-821-0951/微信 atstudy-51

jenkins 实现测试用例并发运行

◆作者：刘晓佳

我们了解了 robotframework 编写自动化测试用例的方法，了解了如何将用例在 jenkins 上运行。但是，随着用例的增多，传统的 pybot/robot 命令运行测试用例会耗费大量的时间，这就慢慢的成为了一个苦恼的问题。

那么，在 jenkins 上如何实现用例的并发运行呢？！我们需要认识的是 jenkins 只是一个持续集成的自动化工具。在 jenkins 服务里，我们可以通过调用 shell 脚本或 python 脚本等的方式实现我们的用例运行。

通过 robotframework 要实现用例的并发运行，不得不提一下 pabot 库——robot framewrok 测试并行执行器，可以将一个执行拆分为多个执行并节约测试执行时间。

1.pabot 安装方法

在线安装方法：使用 pip install -U robotframework-pabot?命令即可；

离线安装方法：通过 https://github.com/mkorpela/pabot 地址下载安装压缩包，解压后，使用 setup.py install 命令即可。

2.pabot 使用方法

1）参数列表

参数	说明
--verbose	更多输出
--command [启动 pabot 的实际命令] --end-command	RF 脚本，用于无法直接使用 pybot 的情况

--processes [处理器个数]	使用多少个并行执行程序(默认最大值为 2 个和 cpu 个数)
--testlevelsplit	在测试级别而不是默认套件级别拆分执行。如果 .pabotsuitenames 包含测试和套件，那么这只会影响新套件并仅拆分它们。当 .pabotsuitenames 文件中的套件和测试都保留此标志时，也只会影响新套件并将它们添加为套件文件。
--resourcefile [文件路径]	可以包含用于分配资源的共享变量的文件的指示符。这需要与 pabotlib 选项一起使用。资源文件语法与 Windows ini 文件相同。其中一个部分是一组共享的变量。
--pabotlib	启动 PabotLib 远程服务器。这可以在并行测试执行之间实现锁定和资源分配。
--pabotlibhost [主机名]	PabotLib 远程服务器的主机名（默认为 127.0.0.1）
--pabotlibport [端口]	PabotLib 远程服务器的端口号（默认为 8270）
--ordering	可以选择从文件中给出执行顺序
--suitesfrom [output.xml 路径]	可选择从 output.xml 文件中读取套件。失败的套件将首先运行，运行时间较长的套件将在较短的套件之前执行。
--argumentfile[整数] [参数文件路径]	使用多个参数文件选项运行相同的套件

2）使用方法

基本使用方法，如：pabot --processsess 2 /home/robotframwork-test。使用 2 个并行执行/home/robotframework-test 目录下的测试用例。

3.pabot 使用场景模拟

1）如何多并发执行测试用例

这是我们选择使用 pabot 的最原始和基本需求，使用--processess 参数即可满足，参考上述”基本使用方法”。

2）如何重复测试失败测试用例

在没有选择使用 pabot 之前，我们使用 pybot 运行用例时，可以使用--rerunfailed 参数读取 output.xml 文件中失败的用例重新执行。命令如：pybot --rerunfailed ./output.xml。当我们使用 pabot 并发运行之后，我们同样可以调用 pybot 的--rerunfailed 参数，重新运行失

败用例。参考命令如：pabot --processesses 2 pybot --rerunfailed /home/robotframework-test

3）如何控制执行顺序

pabot 默认通过 pabotsuitenames 文件控制执行顺序.pabotsuitenames 文件结果如下图 1 所示。前 4 行为 pabot 运行时生成的相关信息，其后为运行的用例套件顺序。如果我们想控制用例执行顺序，可通过--odering [文件名]参数来控制，例如：pabot --ordering file，file 为文件名。file 里的内容前 4 行为空，从第 5 行开始编辑，格式如.pabotsuitenames。

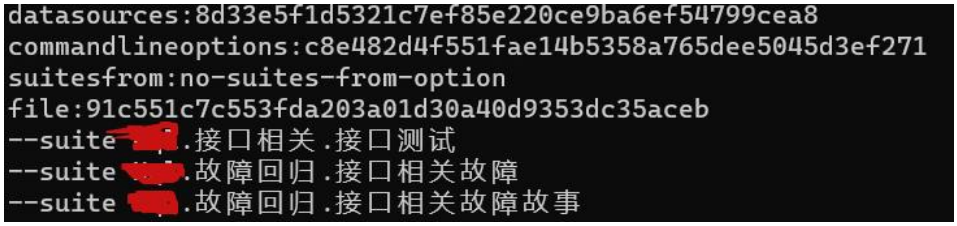


图 1 pabotsuitenames 文件内容

图 2 编辑的 file 文件内容

通过 pabot －ordering file 运行用例，可看到执行顺序如下图 3 所示（ID 号表明执行顺序）。由下图可见，执行顺序于图 2 中指定的顺序一致。

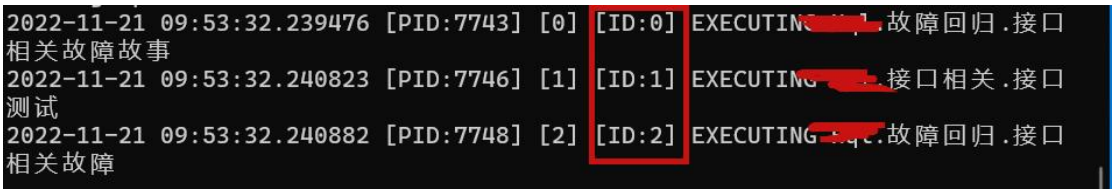


图 3 指定 file 文件控制执行顺序

此外，还可以并添加#WAIT 标志，等待前面的用例执行完后再执行#WAIT 后的用例，参考格式如下图所示。

```
--suite Directory 1 Name.Suite A Name
--suite Directory 1 Name.Suite B Name
--suite Directory 1 Name.Suite C Name
#WAIT
--suite Directory 2 Name.Suite A Name
--suite Directory 2 Name.Suite B Name
```

图 4 #WAIT 参数使用方法

4）如何使用非默认的 pybot/robot 启动命令

有的测试环境可能存在不止一个 pybot/robot 命令，例如：python2 和 python3 并存，且都安装了 robotframework。环境默认使用 python2 语言。当使用 pabot pybot/robot….时，默认使用的时 python2 的 pybot/robot。那么如何使用 python3 的 pybot/robot 呢？

这个时候，使用--command [自定义 pybot/robot 命令] －end-command 即可满足我们的要求。例如：pabot --command python3 -m robot --end-command --processes 2。

4.总结

本文简单介绍了 pabot 的使用方法和一些使用场景。使用 pabot 可以提高我们 robotframework 测试用例的执行效率，但是指的注意的是：pabot 是以测试套件为单位并行运行的。因此可能存在如果用例套件的用例分布不均（比如 A 套件 100 个用例，B 套件 10 个用例），那么用例少的套件则会早早执行完，资源空置无法合理利用。这个时候就需要我们拆分用例套件或均匀化用例，提高执行效率和资源使用率。此外，用例并发更加关键的一点是要去除用例之间的耦合和相互依赖性，避免因为用例之间的顺序依赖导致用例运行失败。

测试用例规范应包括哪些内容

◆ 作者：桃子

最近就测试工作的一些文档进行了总体的梳理总结，关于测试用例规范这方面有一些分享。

今天我们来讲讲测试用例规范，首先什么场景下需要有测试用例规范呢？测试用例规范的内容应该涵盖哪些方面呢？

一 测试用例规范产生的条件

一般新成立的测试部门，需要开展测试工作的情况下需要有一些指导性文件，测试用例规范就是用于指导测试用例编写流程及更改的文件，目的为团队成员在日常工作中开展提供统一依据和标准。

也就是说开展测试工作之前，就应该有这一份文档指导测试用例编写。

二 测试用例规范应包括哪些内容？

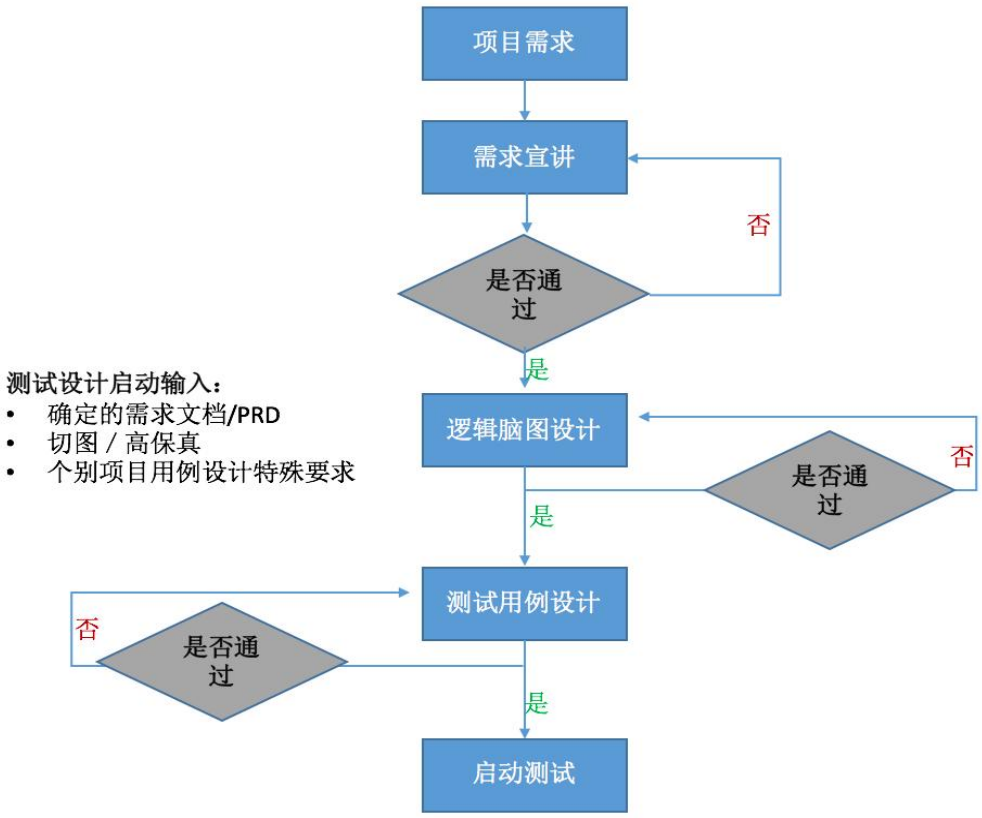
- 规范的范围
- 编写流程
- 编写要求
- 编写要素的定义
- 用例版本维护的原则
- 测试用例模板

1.首先明确我们这个规范的范围

如针对单元测试、接口测试、功能测试等方向，就需要有针对这三个方向的对应准则。

2.编写流程

最好画一个关于测试用例编写的流程图，这样看文档的人直接看流程图就能了解到测试用例的工作流程是什么样，如下图：



3.编写要求

用于说明单元、接口、功能测试用例的编写要求是什么，如果没有单元测试，这部分内容可以不写，拿功能测试要求说明：

- 功能测试用例颗粒度原则
- 功能测试用例准确性原则
- 功能测试用例的可维护性和可移植性原则

• 功能测试用例覆盖度原则

每一项具体的原则是什么根据自己的公司实际情况而定。

4.编写要素的定义分别是什么

这部分内容是指测试用例的组成要素，包括 功能模块名称、测试用例编号、测试用例标题、前置条件、测试用例步骤、期望结果、测试用例优先级（P0，P1，P2 级）、用例类型（功能、UI 等）、编写人、执行人、测试结果、bug 编号、创建日期、测试日期、是否需求变更新增、新增日期、备注）。

针对每一个要素应具体说明如何使用，如测试用例编号的格式及规则应是什么样的，如 SZDZ-001。

再或者用例优先级的说明：

分 P0、P1、P2、共 3 级，各级描述和用例占比如下：

优先级	描述	Case 比例	说明
P0	即冒烟测试用例，覆盖产品主要流程，该部分用例执行失败会导致多处重要功能无法使用，需求提测初期可作为确定当前版本是否可测的用例	20%	开发自测用例范围
P1	覆盖产品的基本功能点及重要的异常流程，此部分和P0测试用例执行通过后能够保证产品的功能是基本稳定的	20%	提交客户集成／验收之前的最终版本需要100%保证P0,P1测试用例的覆盖
P2	详细覆盖产品功能区域测试，包含字段边界值、异常配置等校验；同时包含产品易用性，GUI或非场重要的异常流程及出发条件较特殊的场景	60%	

[说明]各级用例占比上下浮动控制在±3%以内。

5.用例版本维护的原则

说明测试用例维护触发的条件应是什么样：

1）版本号变更，什么情况下大版本变更，什么情况下小版本变更。

2）标注规则：更新或新增的测试用例，以黄色底色 / 蓝色字体作为标识。

3）文档留存原则：所有版本用例均不需要删除，在存储路径中加入修改后版本即可。

6.测试用例模板：最后附上用例模板供参考

测试结果	数量	比率	
Passed/Run	0	#DIV/0!	
Failed/Run	0	#DIV/0!	
Blocked/Run	0	#DIV/0!	由于Bug或测试数据不全导致的Block
NA/Run	0	#DIV/0!	由于本期不做或无测试环境导致的不可测试项
Run/Total	0	#DIV/0!	
Total	0		有效Case总数

序号	模块	子模块	测试类型	测试目的	前提条件	测试步骤	期待结果	备注	优先级	测试结果	bug ID	Bug详细信息	测试人员	测试时间	测试版本
			入口												
			UI检查												
			业务流程实现检查												
			兼容性测试												
			性能测试												

环境搭建的趟坑之旅——从 Django 到 VUE

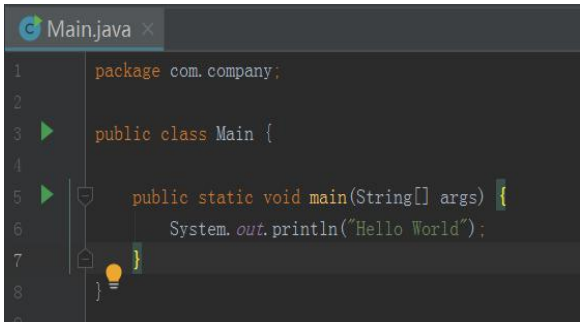
◆作者：Bella

写点代码解决下工作中的小问题已经成为现在测试人的日常，可是如果需要做个有前后端的“系统”，该怎么弄呢？公司是内网环境，所有东西都需要自己从外网找然后打包压缩发到内网，每次发送还有大小限制，经常遇到的问题是花了两天安装几十个插件之后，发现有一个依赖包无论如何都无法安装，八成是不支持当前系统或者和什么内部的软件冲突了。为了能够顺利完成“工具开发”任务，最后决定和开发人员统一环境，虽然权限比人家少了很多，有些包仍然需要从外网搬运，但是至少搭建完了能用，遇到问题有专家能支持。

很久之前写过一些 Java，但是自从试了一下 Python，就果断抛弃了 Java，而后一直用的也是 Python，唯一的原因就是“方便”，不用想 a=1 前面是 int 还是 string，安装 django 只是一个 pip install 的事情。但是为了收敛技术栈，需要换回 Java，于是开启了更换开发环境的探索之旅。现在比较火的就是 Springboot+VUE，于是开始上网撸教程，“从 0 开始 xxx”、“4 小时学会 xxx”、“1 天搞定 xxx”……看完之后的感觉就是，这真的是 0 基础吗？老师上来就说“我默认大家已经有相关基础了”、“不需要我说怎么安装怎么使用哈，这都是基本操作”……算了，我还是自己摸索吧。过程比较曲折，想看 tips 的请直接跳到文末。

首先就是 IDE，之前一直用的是 Pycharm，现在要改到 Java，在 VSCode 和 IDEA 之间摇摆了一下，决定使用 IDEA（因为只找到了这个安装包），带来的问题就是找网上教程的时候，选择范围缩小了一点，尤其是有关配置类的，需要挨个摸索一下。

太久不写 Java，已经不记得多少了，赶紧先新建个项目，跑个 Hello World，写惯了 print(“Hello World”)，竟然已经不记得 print 前面要加 System.out 了，任重道远啊。由于本地有 JDK1.8，直接执行就可以成功。




```
Hello World

Process finished with exit code 0
```

然后就是安装 NodeJS，管开发要了个安装包，一路 next，并没有什么意外，NPM 是跟随 NodeJS 一起的包管理工具，无需单独安装，完成后到 cmd 里执行 node -v 和 npm -v 看一下是不是安装成功了。

```
C:\Users>node -v
v10.14.2

C:\Users>npm -v
6.4.1
```

输入 vue -V 之后显示了版本号，但是教程里是@vue/client <version>，看起来不太一样呀，还是再摸索一下。

```
C:\Users>vue -V
2.9.6
```

于是去搜了个 vue client 安装教程，都说 npm install -g @vue/cli，看着就没有网络连接，应该成功不了。

```
npm ERR! network In most cases you are behind a proxy or have bad network
```

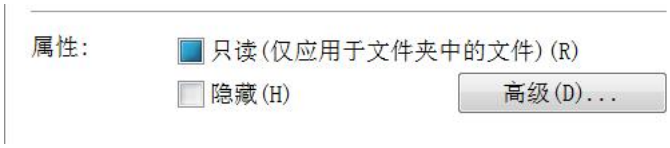
又搜了个离线教程，说要在外网下载完后，把文件夹传到内网，结果发现内网是 win7，外网的都是 win10 和 win11，果然放进来还是不能用。最后还是决定找个别人的库连上试试能不能安装，按照教程来说，出现版本号就是成功了，这么看起来仍然不行。

```
npm ERR! path D:\software\nodejs\node_modules\vue
npm ERR! code EISDIR
npm ERR! errno -4068
npm ERR! syscall open
npm ERR! EISDIR: illegal operation on a directory, open 'D:\software\nodejs\node
_modules\vue'

npm ERR! A complete log of this run can be found in:
npm ERR!     D:\software\nodejs\node_modules\_logs\2022-12-30T02_02_58_595Z-debu
g.log
```

打开 log 看看，发现报错是 not permitted。

```
20019 warn rollback Rolling back kind-of@3.2.2 failed (this is probably harmless): EPERM: operation not permitted, scandir 'D:\software\nodejs\node_modules
20020 warn rollback Rolling back is-data-descriptor@1.0.0 failed (this is probably harmless): EPERM: operation not permitted, scandir 'D:\software\nodejs\n
```

怀疑是文件夹权限问题，打开看看是带只读属性的，去掉只读再试一次。



还是报错，看了下 log，应该是缺依赖包了。

```
20021 warn node-fetch@2.6.7 requires a peer of encoding@^0.1.0 but none is installed. You must install peer dependencies yourself.
20022 warn ws@7.5.9 requires a peer of bufferutil@^4.0.1 but none is installed. You must install peer dependencies yourself.
20023 warn ws@7.5.9 requires a peer of utf-8-validate@^5.0.2 but none is installed. You must install peer dependencies yourself.
```

安装完了所有的依赖包，再试试安装 vue/cli 脚手架，仍然报一样的错，在非常没有头绪的时候打算先创建个工程，结果突然发现可能是版本问题。

```
D:\automation\vue_demo>vue create portal

  vue create is a Vue CLI 3 only command and you are using Vue CLI 2.9.6.
  You may want to run the following to upgrade to Vue CLI 3:

  npm uninstall -g vue-cli
  npm install -g @vue/cli
```

去搜了下 VUE CLI 2 的脚手架安装方式，果然跟 VUE CLI 3 非常不一样，需要先执行 npm install -g @vue/cli-init。

```
D:\software\nodejs\node_modules>npm install -g @vue/cli-init
npm WARN deprecated vue-cli@2.9.6: This package has been deprecated in favour of
 @vue/cli
npm WARN deprecated coffee-script@1.12.7: CoffeeScript on NPM has moved to "coff
eescript" (no hyphen)

> metalsmith@2.5.1 postinstall D:\software\nodejs\node_modules\node_modules\@vue
\cli-init\node_modules\metalsmith
> node metalsmith-migrated-plugins.js || exit 0

+ @vue/cli-init@5.0.8
added 244 packages from 203 contributors in 107.919s
```

然后执行 npm install webpack

```
D:\automation\vue_demo>npm install webpack
npm WARN saveError ENOENT: no such file or directory, open 'D:\automation\vue_de
mo\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.

npm WARN enoent ENOENT: no such file or directory, open 'D:\automation\vue_demo\
package.json'
npm WARN vue_demo No description
npm WARN vue_demo No repository field.
npm WARN vue_demo No README data
npm WARN vue_demo No license field.

+ webpack@5.75.0
added 77 packages from 124 contributors and audited 77 packages in 21.419s
found 0 vulnerabilities
```

到工程目录下执行 npm init webpack helloworld。

```
D:\automation\vue_demo>vue init webpack helloworld

   vue-cli · Failed to download repo vuejs-templates/webpack: getaddrinfo ENOTF
OUND github.com github.com:443
```

目测是超时了，看攻略说是要从外网下一个然后传进来用，因为还不知道后面有多少坑，所以还是决定改成 VUE 3，先把 2 卸载了，执行 npm uninstall -g vue-cli。

```
D:\automation\vue_demo>npm uninstall vue-cli -g
up to date in 0.063s
```

发现还是能查到版本号，按照攻略依次执行 npm config ls -l， 删除.npmrc 文件。

```
C:\Users>npm config ls -l
```

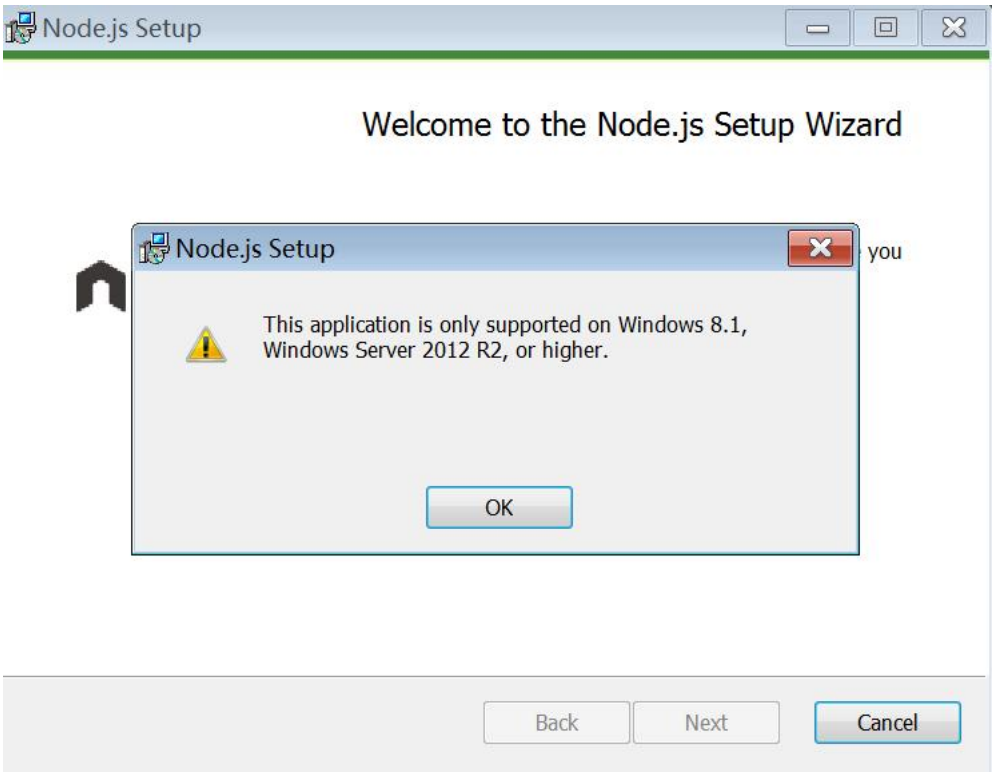
执行 where vue，找到 vue 和 vue.cmd，全都删除。

```
C:\Users>where vue
```

再次执行 vue -V，不显示版本号，说明删除成功了。

```
C:\Users>vue -V
'vue' 不是内部或外部命令，也不是可运行的程序
或批处理文件。
```

重装 npm install -g @vue/cli 之后发现仍然是 VUE CLI 2，于是决定升级下 nodejs，下了 18.12，结果发现 win7 能够支持的最高版本是 13.14。



退出去重来，13.14 没问题，npm 没问题，vue 没问题。

```
+ @vue/cli@5.0.8
added 837 packages from 493 contributors in 222.307s
```

新建项目时候报了另外一个错，于是按照在 win7 上安装 16.3.0 的说明再来一次。

```
D:\automation\vue_demo>vue create helloworld
You are using Node v13.14.0, but this version of @vue/cli requires Node ^12.0.0
|| >= 14.0.0.
Please upgrade your Node version.
```

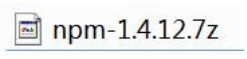
下载压缩版本的 node-v16.3.0-win-x64.7z，解压后覆盖原来 13.14 文件夹内容，添加环境变量 NODE_SKIP_PLATFORM_CHECK=1，查看版本仍然为 13.14，更换为 node-v16.3.0-win-x64.zip 后版本正确。

```
C:\Users>node -v
v16.3.0
```

但是现在的问题变成没有 npm 了，先查一下确认当前 node 对应的 NPM 版本 https://nodejs.org/zh-cn/download/releases/。

Version	LTS	Date	V8	npm	NODE_MODULE_VERSION[1]			
Type versions of Node.js or npm to search (e.g. 'Node.js 14.17.5' or '6.14.14' ...)								
Node.js 16.5.0		2021-07-14	9.1.269.38	7.19.1	93	下载	更新日志	文档
Node.js 16.4.2		2021-07-05	9.1.269.36	7.18.1	93	下载	更新日志	文档
Node.js 16.4.1		2021-07-01	9.1.269.36	7.18.1	93	下载	更新日志	文档
Node.js 16.4.0		2021-06-23	9.1.269.36	7.18.1	93	下载	更新日志	文档
Node.js 16.3.0		2021-06-03	9.0.257.25	7.15.1	93	下载	更新日志	文档
Node.js 16.2.0		2021-05-19	9.0.257.25	7.13.0	93	下载	更新日志	文档
Node.js 16.1.0		2021-05-04	9.0.257.24	7.11.2	93	下载	更新日志	文档
Node.js 16.0.0		2021-04-20	9.0.257.17	7.10.0	93	下载	更新日志	文档
Node.js 15.14.0		2021-04-06	8.6.395.17	7.7.6	88	下载	更新日志	文档
Node.js 15.13.0		2021-03-31	8.6.395.17	7.7.6	88	下载	更新日志	文档
1 ... 4 5 6 7 8 9 10 ... 72								

然后到 NPM 的下载界面去下载一个对应版本的，但是并没有找到，可以先随便下载一个，然后再升降级 https://registry.npmmirror.com/binary.html?path=node/npm/，解压复制到 nodejs 的安装目录下，使用 npm -v 查看版本号是可以成功的。




```
C:\Users>npm -v
1.4.12
```

使用时会报版本过低，无法使用 npm install -g npm@7.15.1，如果想升级到支持的最新版本的话，需要升级 NPM，在其他路径安装更高版本的 node，用 node_module/npm 文件夹覆盖当前的文件夹，再执行升级命令，还是有兼容性的问题，问了有经验的前端同事，内网环境可能有冲突，镜像可能不支持，和大佬商量了下决定退回到 VUE2。

```
C:\Users>npm login
npm WARN npm npm does not support Node.js v16.3.0
npm WARN npm You should probably upgrade to a newer version of node as we
npm WARN npm can't make any promises that npm will work with this version.
npm WARN npm Supported releases of Node.js are the latest release of 6, 8, 9, 10
, 11.
npm WARN npm You can find the latest version at https://nodejs.org/
npm WARN npm npm does not support Node.js v16.3.0
npm WARN npm You should probably upgrade to a newer version of node as we
npm WARN npm can't make any promises that npm will work with this version.
npm WARN npm Supported releases of Node.js are the latest release of 6, 8, 9, 10
, 11.
npm WARN npm You can find the latest version at https://nodejs.org/
```

删除所有的内容，尤其是要执行 npm uninstall -g npm，否则重装之后会报错。

```
C:\Users>npm login
npm ERR! Cannot read property 'resolve' of undefined

npm ERR! A complete log of this run can be found in:
npm ERR!     C:\Users\zhangmiaoyfz\AppData\Roaming\npm-cache\_logs\2023-01-04T02
_27_04_085Z-debug.log
```

整个重来之后终于成功了，由于 VUE3 和 VUE2 有很大区别，包括语法，如果有机会还是要试试 VUE3 的。VUE 删除如果不干净的话，是无法降级的，需要先执行 npm uninstall vu-cli -g，然后手动将所有相关文件夹内包含 vue 字样的文件删除，环境变量如果有配置也最好删除，重装重新打开 cmd，执行 npm i vue-cli -g，可见版本已降级。

```
D:\automation\vue_demo>npm ls vue -g
`-- @vue/cli@5.0.8
  `-- vue@2.7.14
```

```
+ vue-cli@2.9.6
updated 1 package in 24.344s
```

新建一个 VUE2 的项目，用 vue init webpack <project name>，仍然报错，这个应该就是缺少依赖包，还是需要从外部下载传进来的。

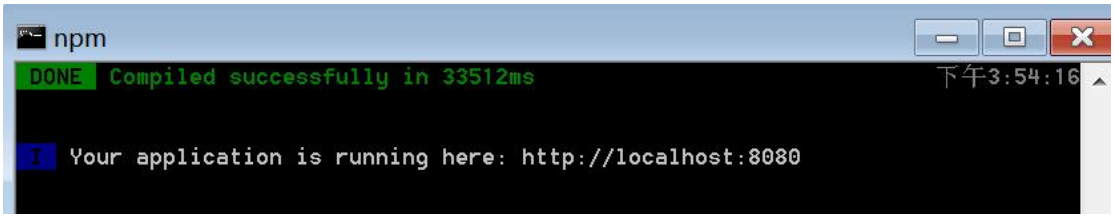
```
D:\automation\vue_demo>vue init webpack helloworld

   vue-cli · Failed to download repo vuejs-templates/webpack: getaddrinfo ENOTF
OUND github.com
```

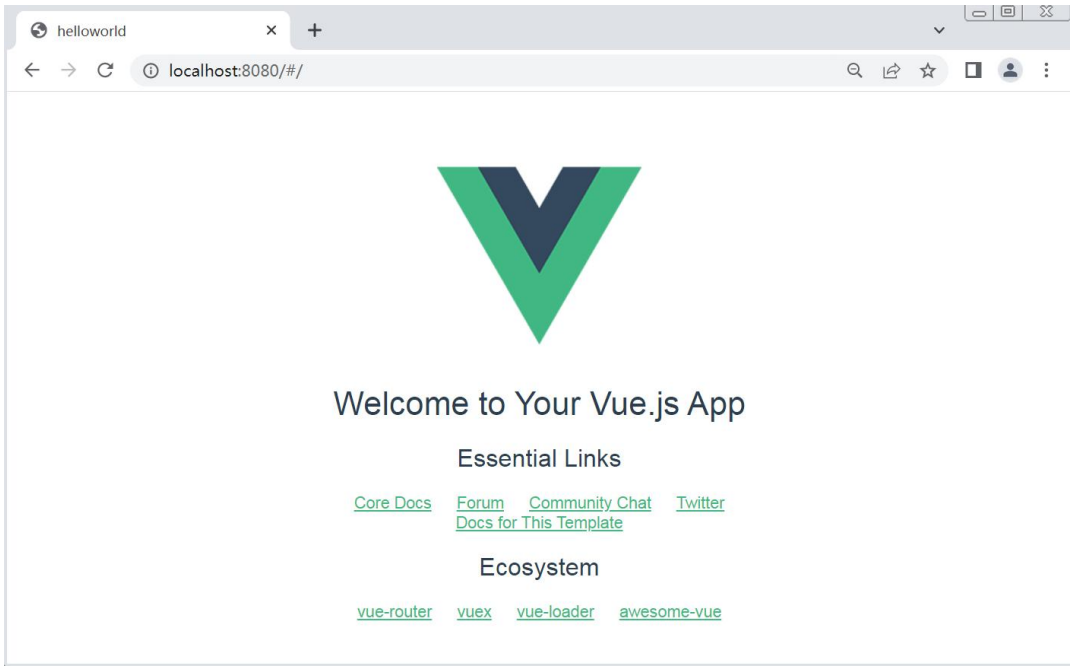
让同事给穿了个 webpack 的包，在 C:\user\xxx\下创建一个名为.vue-templates 的文件夹，把 webpack 整个复制进去，然后 cd 到工程目录，执行 vue init webpack <project name> --offiline，终于成功了！

```
# Project initialization finished!
# ========================

To get started:

  cd helloworld
  npm run dev

Documentation can be found at https://vuejs-templates.github.io/webpack
```

```
D:\automation\vue_demo>cd helloworld

D:\automation\vue_demo\helloworld>npm run dev

> helloworld@1.0.0 dev D:\automation\vue_demo\helloworld
> webpack-dev-server --inline --progress --config build/webpack.dev.conf.js
```

等一会儿，然后在浏览器打开。





至此就算大功告成啦，后面就是总结这一路的排坑指南了：

1.Java 不用重装；

2.先在官网查好自己的系统支持的 nodejs 版本再安装；

3.能选择 VUE3 直接 3，如果不行退而求其次用 VUE2.7，语法会更接近 3，方便以后升级；

4.如果需要删除重装，一定要删干净，安装目录和 cache 都要删；

5.更改安装路径后，记得改环境变量；

6.VUE2 和 3 的命令不一样，现在网上的教程有对有错，如果不成功就换个教程。

拓展学习

[1] 从 0-1 实现 Django 项目搭建

https://www.atstudy.com/course/1011202

[2] 基于 Django 的企业级自动化测试平台开发

https://www.atstudy.com/course/1010682

谨防并发测试的坑

◆ 作者：刘晓佳

一、案情描述

收到这么一个需求：存在一个数据库查询功能接口，需要完成 1000 个条件语句的查询，并将查询结果与原始数据库（如 es）的查询结果对比，从而判定该功能接口是否正常，且正确可用。

第一次测试设计：读取文件中的 1000 个查询条件，顺序执行。但由于查询数据量较大，单线程顺序完成 1000 次查询耗时较长，效率太低——抛弃。

第二次测试设计：使用并发查询，多线程并发提高工作效率，节省了大大的时间。但将输出的 1000 个查询结果与原始数据库查询结果对比时，发现某些语句差异较大，为什么？

二、抽丝剥茧

为何功能接口查询结果与原始数据查询结果差异性大？

经过排查，问题出在查询语句对比时匹配错位——使用简单并发查询，输出并不严格按照读入顺序。换句话说：读入 1、2、3、4、5 个顺序查询条件，并发查询输出顺序可能时 2、1、3、4、5，也可能时 4、5、2、1、3。

那么，如何在编写测试脚本时，让并发输出严格按照读入顺序输出呢？

三、亡羊补牢

从 Python3.2 开始，标准库为我们提供了 concurrent.futures 模块，它提供了 ThreadPoolExecutor 和 ProcessPoolExecutor 两个类，可以帮助我们更好地完成并发设计。本文我们不过深地讲述如何使用这些 python 类，只简单讲述如何解答我们的核心问题——“如何在并发时让结果严格按照输入顺序输出”。

1.线程池 ThreadPoolExecutor

如下所示，为 ThreadPoolExecutor 的基本使用方法，打印输入的 list 列表中的数字。ThreadPoolExecutor()初始化，创建线程池，最多 2 个线程并发运行；通过 submit 调用子函数（打印输入数字）；最终通过 as_completed 等待所有任务完成后，通过 result 收集返回结果。

```
# -*-coding:utf-8 -*-
from concurrent.futures import ThreadPoolExecutor,as_completed

# 子函数，打印输入数字
def print_num(num):
    return num

list = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

###############使用 ThreadPoolExecutor###################
################并发执行，非顺序返回################
# 创建线程池子
# 设置最多 2 个线程运行，其他等待
executor = ThreadPoolExecutor(max_workers=2)
all_task = [executor.submit(print_num, (num)) for num in list]
result2_list = []
for future in as_completed(all_task):
    data = future.result()
    result2_list.append(data)
print("并发执行，非顺序返回：  "+ str(result2_list))
##############################################
```

如下图所示为上述使用 ThreadPoolExecutor 并发运行的结果，由结果打印可知，输出并非严格按照列表输入值顺序输出。由此可以预见，当我们简单使用并发时，我们的结果可能并不是我们认为的与“顺序”输出。

```
并发执行，非顺序返回：  [3, 1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

那么，如何改造呢？请关注 map 方法。使用 map 方法，无需提前使用 submit 方法，map 方法与 python 标准库中的 map 含义相同，都是将序列中的每个元素都执行同一个函数。

```
###############并发执行，顺序返回################
# 创建线程池子
# 设置最多 2 个线程运行，其他等待
executor = ThreadPoolExecutor(max_workers=2)
result1_list = []
for result in executor.map(print_num, list):
    result1_list.append(result)
print("并发执行，顺序返回：  " + str(result1_list))
##############################################
```

上面的代码就是对 list 的每个元素都执行 print_num 子函数，并分配各线程池。

```
并发执行，顺序返回：  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

由此可以看到，执行结果与上面的 as_completed 方法的结果不同，输出顺序和 list 列表的顺序相同。

2.进程池 ProcessPoolExecutor

使用进程池 ProcessPoolExecutor 处理并发输入，按序输出问题，同 ThreadPoolExecutor 一样，使用 map 方法即可解决。但是值得注意的是，注意，在使用多进程时，必须把 多

进程代码写在?if __name__ == '__main__' 下面，否则异常，甚至报错

“concurrent.futures.process.BrokenProcessPool: A process in the process pool was terminated abruptly while the future was running or pending.”

```
##########################使用 ProcessPoolExecutor######################
###############并发执行，顺序返回################
# 创建进程池子
# 设置最多 2 个进程运行，其他等待
if __name__ == '__main__':
    executor = ProcessPoolExecutor(max_workers=2)
    result2_list = []
    for result in executor.map(print_num, list):
        result2_list.append(result)
    print("ProcessPoolExecutor 并发执行，顺序返回：  " + str(result2_list))
##############################################
#########################################################################
```

3.还存在一点“笨方法”

除了上述所说的 ThreadPoolExecutor 和 ProcessPoolExecutor 使用 map 方法让结果顺序输出外，我们还可以使用一些笨方法。例如，使用并发运行程序（并发方法不仅限于 ThreadPoolExecutor 和 ProcessPoolExecutor，还可以使用 threading 等等）sort 方法对输出结果排序。就上述案例而言，不使用 map 可以如下改造：

```
###############并发执行，非顺序返回################
# 创建线程池子
# 设置最多 2 个线程运行，其他等待
executor = ThreadPoolExecutor(max_workers=2)
all_task = [executor.submit(print_num, (num)) for num in list]
result2_list = []
for future in as_completed(all_task):
    data = future.result()
    result2_list.append(data)
result2_list.sort()
```

```
print("ThreadPoolExecutor 并发执行，非顺序返回：  "+ str(result2_list))
##################################################
```

最终，输出结果符合我们的要求。

```
ThreadPoolExecutor并发执行，非顺序返回：  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

四、结案陈词

ThreadPoolExecutor 多线程并行执行任务，可以共享当前进程变量，但缺点也很致命，但其实仍然最多只能占用 CPU 一个核。如果指定的任务和线程数不恰当(比如一个任务很短，线程数量很多，导致线程频繁调用回收)，那么效率还不如单线程。ProcessPoolExecutor 可以使用多核进行计算，但缺点就是进程之间共享数据就比较麻烦，消耗更多的内存。

本文的主要目的是帮助大家认识：并发好用，但使用需谨慎。谨防一不小心落入并发的坑，使用 map 方法可以帮助大家快速地完成测试结果地有序输出。

如何跟踪界面弹窗的接口调用和传参

◆作者：刘晓佳

一、前情回顾

在之前的文章里（如《如何通过界面元素准确找到使用得后端接口》、《如何“顺藤摸瓜”对 web 应用 bug 进行初步分析》）我们了解了如何简单判断界面出现的问题是属于后端接口故障还是前端故障，了解了如何通过前端提示或界面元素如何快速找到绑定的后端接口。我们在学习到这类知识后获得了简单的调试和分析能力，已经能够独立分析一些故障原因。但是问题总是层出不穷的，习得的方法并不能一劳永逸……

二、故事继续

有这么一个场景：当你点击界面某个按钮或者某个图标，开启了一个新标签页。但是，你发现新标签页的某些选项设置与需求不一致。比如：你期望新标签页上的搜索框，设定的搜索时间是上级页面选择的 7 天，但新标签页上搜索框搜索时间范围设定确是 1 天。

这个时候，你反复尝试、清楚浏览器缓存、仔细观察每个步骤，避免人为因素导致测试结果偏差。但，无数次尝试后你断定：的确是个 bug。

你想直接告诉前端开发同事：xxx 搜索框传入搜索时间范围不正确。但你按捺住了冲动，你想搜集更多的证据证明这的确是前端的 bug。毕竟，有理有据，说服人都能理直气壮。

可是，弹出新标签页不同于同一页面变化，要怎么跟踪从上级页面传入下级页面的参数呢？或者说，怎么能够跟踪新页面的接口调用呢？

三、主线任务

如何跟踪弹出页面的接口调用以及上级页面传入的参数呢？

光说不练假把式。还是以 51testing 网站为例进行讲解~

如下图所示为 51testing 界面，点击“最新更新”栏目下的“行业资讯”，会弹出新标签页打开链接，访问详细内容。

图 1 51testing 首页

我们首先来解决我们本次讨论的核心问题：如何跟踪弹窗开启的新标签页调用的后端接口？

这个问题，可能有人会回答：在新标签页开启浏览器调试模式，切换到“网络”，然后刷新页面。

的确，不能否认这是一个跟踪新标签页调用后端接口的方法。但是，请大家想想，刷新页面查看接口调用和上级页面触发调用接口是完全一样的吗？！

至少，有一种情况不一样。如果新标签页需要接受上级页面传参，且刷新页面该参数值会恢复默认值的情况。一旦使用刷新页面跟踪接口调用，我们有可能给会失去上级页面传入的参数，从而无法真正知晓是否上级页面传入参数有误。

那么，要怎么做呢？你需要掌握以下几个浏览器调试模式小技巧。

1）勾选“网络”>“保留日志”

打开浏览器调试模式，切换到“网络”，勾选“保留日志”。这个时候在上级页面点击图表或按钮跳转时，即可看到上级页面调用的接口。如下图所示，点击“最新更新”栏目下的“行业资讯”，弹出新标签页时，在上级页面调试模式可看到事件触发时调用的接口

http://m6816.talk99.cn/monitor/s?c=e&i=20001818&v=65a28119f8eb5cd040470977326503e3&p=882593729&x=1667542852272，及相应的参数。

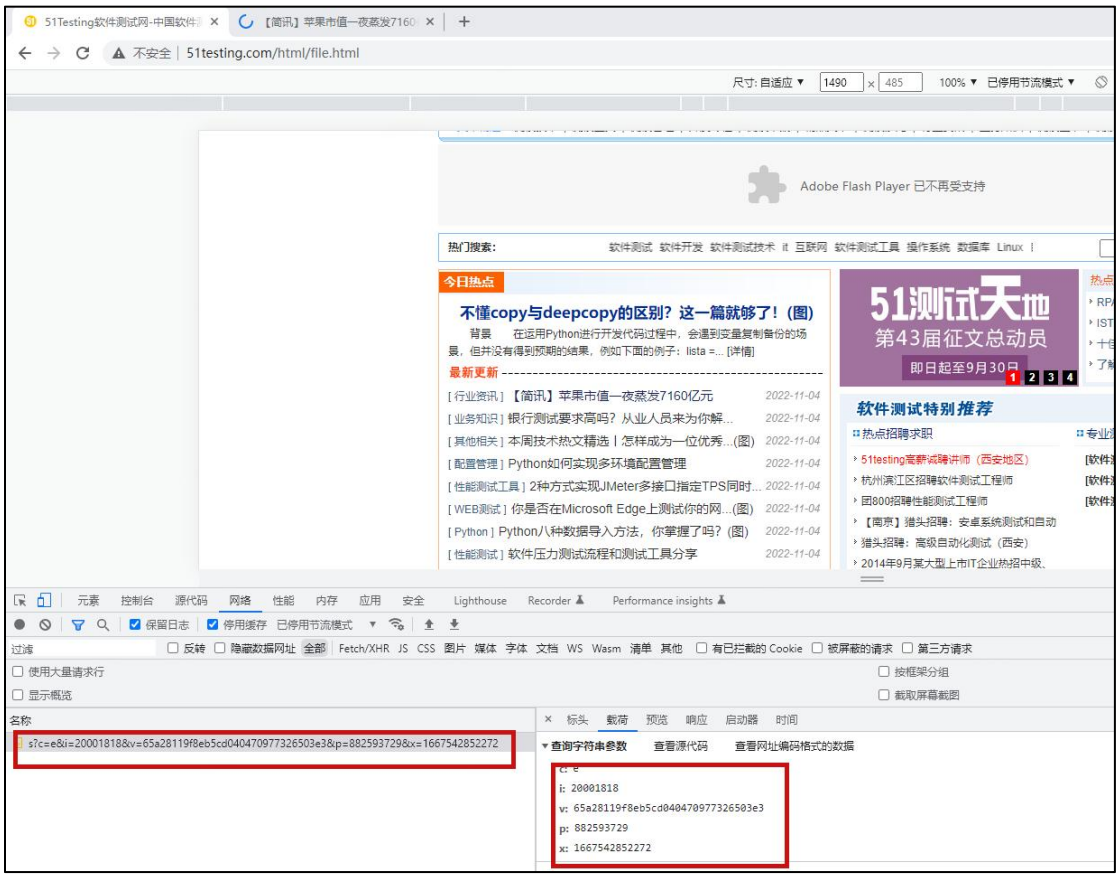


图 2 51testing 上级页面弹出新标签页事件触发时调用的接口

2）勾选“控制台”>“保留日志”

众所周知，调试模式的“网络“面板主要是跟踪接口调用。除了上述 1）所述的方法，我们还可以通过勾选”控制台“>”保留日志“，跟踪弹出窗口新标签页开启时，上级页面的前端日志。设置方式如下图所示。

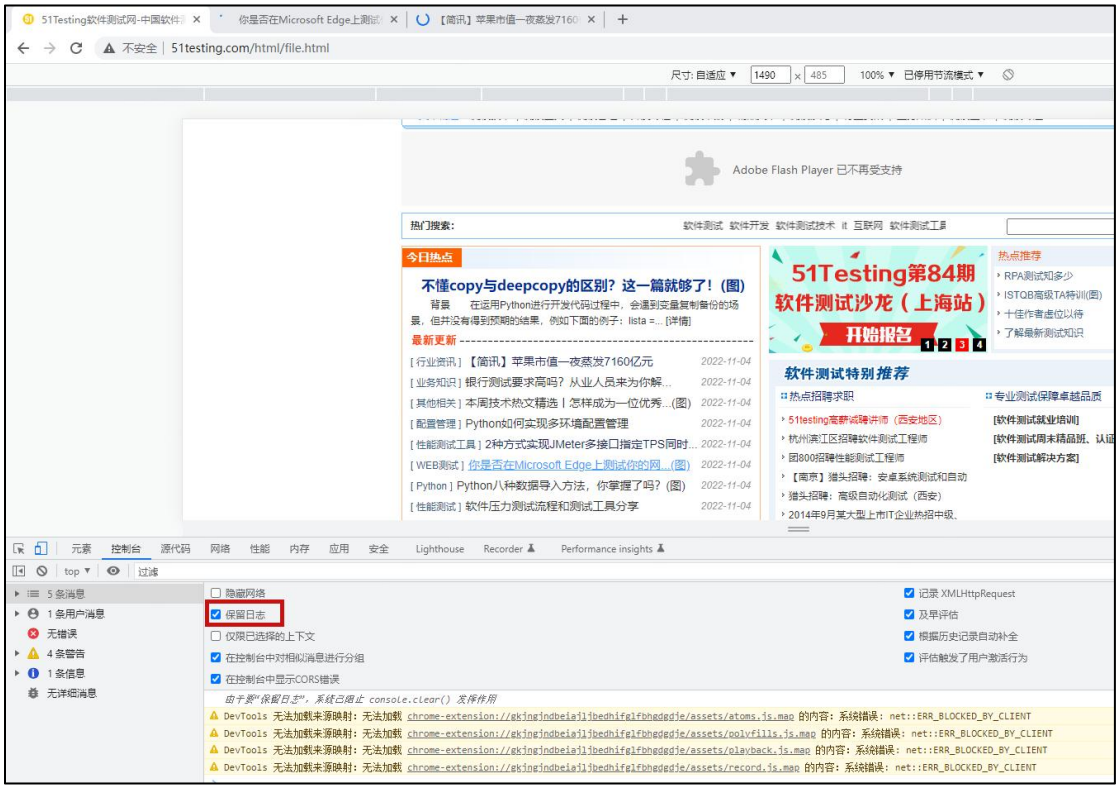


图 3 ”控制台“>”保留日志“设置

由于本案例中，“控制台“未输出相关日志，因此暂不截图演示。

3）开启“为弹式窗口自动打开 DevTools”

我们 1）、2）讲的方法都是捕捉上级页面事件触发时的日志输出，那么对于新页面我们怎么在事件触发时第一时间捕捉发起的接口调用和前端输出呢？！

了解一下“为弹式窗口自动打开 DevTools“设置？！这个设置简直不要太好用。

打开浏览器调试窗口，点击右上方“设置“图标，勾选“为弹式窗口自动打开 DevTools“设置即可。

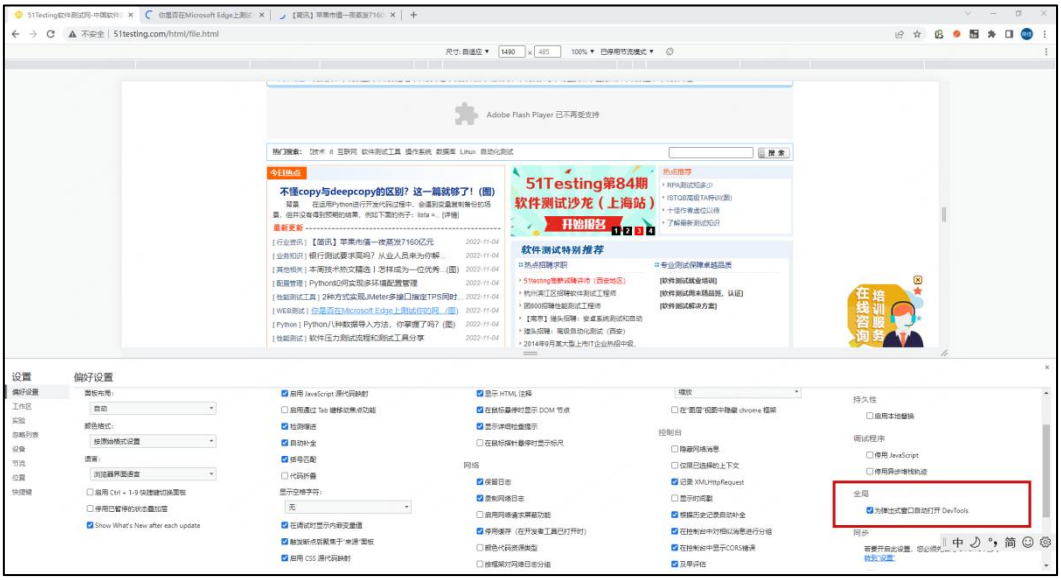


图 4 “为弹式窗口自动打开 DevTools“设置

完成“为弹式窗口自动打开 DevTools“设置后，在上级页面触发弹出窗口开启新标签页时，新标签页会自动打开浏览器调试模式。如下图所示，51testing 的“行业资讯”新标签页调式模式下的网络面板。通过该面板可以及时跟踪新页面调用接口，以及从接口中捕捉上级页面传入的参数。

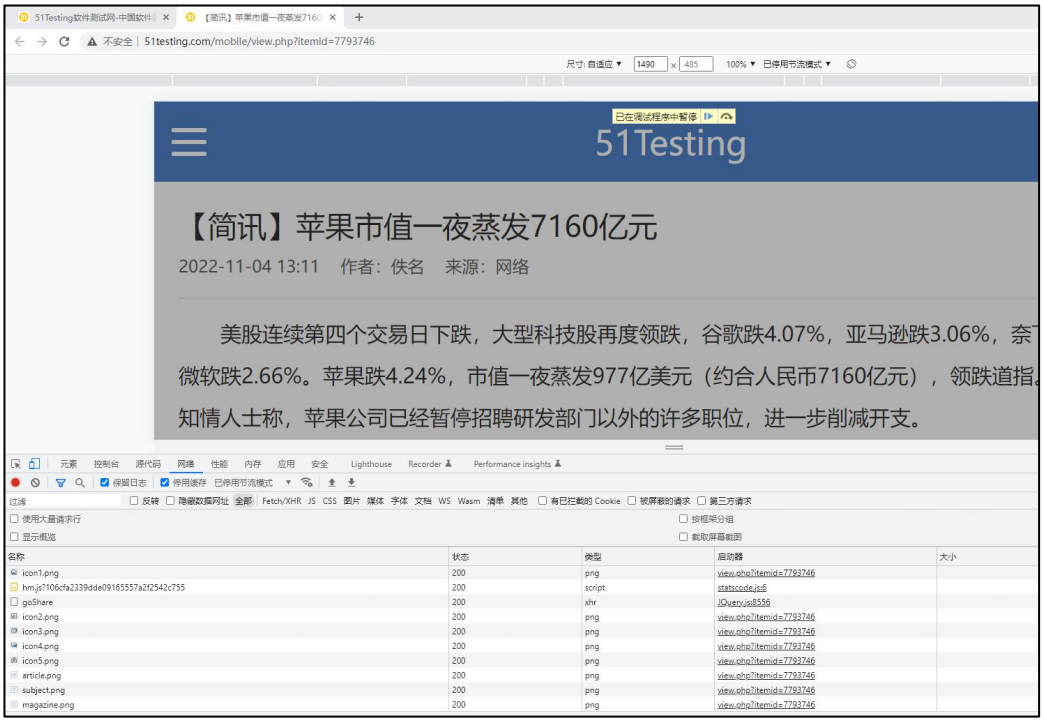


图 5 51testing 的“行业资讯”新标签页调式模式下的网络面板

四、任务交割

通过本文的叙述和简单案例的讲解，我想大家对“如何跟踪界面弹窗的接口调用和传参“有了初步的认识，且掌握了几个小技巧。希望这些小技巧能帮助你更好地完成测试工作。记住：我们的目标是——没有蛀牙！哦，不。是测试工作更专业！

从 0 到 1，保姆级 MobSF 搭建及使用

◆ 作者：奕然

1、引言

时间如梭，梭梭催人老。

一转眼，已经快一个月没有更文了，只因为这段时间，我 90%的时间只投入了两件事：

①产品的安全测试；

②以队长身份参加我厂组织的安全渗透活动。

最后的比赛结果结果，当然是不负众望，在总厂与子长的 18 支参赛队伍(70+人数)中，我(Carl_奕然)荣获个人第二名，团队第二的成绩。

因为习惯了获奖的感(赶)觉(脚)，所以，对我来说，相对于奖牌与名次，更喜欢我的 Boss 们开心的样子。

毕竟，月底的 KPI、年终总结。你懂得..

当然，在参与项目这段时间，我也有了新的成长，对渗透测试，也有了新的提升。

所以，今天，我也准备分享移动 APP 的安全渗透测试，在这里，你会学到：

①移动 APP 安全测试的重要性；

②主流的移动 APP 测试工具；

③从 0 到 1 搭建 Mobile Security Framework(俗称 MobSF)；

④搭建过程中的避坑指南

⑤如何使用 MobSF；

⑥测试报告总结及分析。

或许你会有疑问：

在网上搜索一下，就能知道 MobSF 的搭建方法，为什么还要分享？

因为，网上的那些所谓的教程，几乎很难看到非常详细的，并且没有自己从 0 到 1 的亲自实践、及总结。

还有一点，也是我认为要整理并分享的原因，即：如何解决搭建及使用过程中遇到的问题。

看到这里，是不是已经有些期待了呢？

别停下，跟着我的思路，开启今天的课程学习吧。

2、软件安全测试

2.1 什么是信息安全

关于信息安全，现在的互联网时代，无时无刻的不被提醒，例如：

- 移动/联调/电信的短信提示；
- 下载软件时的信息安全提示；
- 你给陌生人转账时的信息安全认证；

说了这么多，那到底什么是信息安全呢？

引用官方的话：信息安全事关国家生产运行、国家经济安全和人民生命财产安全，是网络强国战略、制造强国战略的重要支撑内容。

你看，信息安全的重要性，下到我们的个人身份信息， 上到国家信息战略，

所以，这也凸显出，信息安全的重要性。

既然信息安全这么重要，那如何避免信息威胁呢，信息威胁都有哪些种类呢？

我们接着往下看。

2.2 信息安全威胁

要想保障信息安全，我们就需要先了解，信息威胁的种类都有哪些，

我先上一个表格，这是 2021 年 OWASP TOP 10 的安全漏洞：

等级	漏洞名称	描述
A01	失效的访问控制 (BrokenAccess Control)	从第 5 位上升成为 Web 应用程序安全风险最严重的类别；提供的数据表明，平均 3.81%的测试应用程序具有一个或多个 CWE，且此类风险中 CWE 总发生漏洞应用数超过 31.8 万次。在应用程序中出现的 34 个匹配为“失效的访问控制”的 CWE 次数比任何其他类别都多。
A02	加密机制失效 (Cryptographic Failures)	排名上升一位。其以前被称为“A3:2017-敏感信息泄漏（Sensitive Data Exposure）”。敏感信息泄漏是常见的症状，而非根本原因。更新后的名称侧重于与密码学相关的风险，即之前已经隐含的根本原因。此类风险通常会导致敏感数据泄露或系统被攻破。
A03	注入(Injection)	排名下滑两位。94%的应用程序进行了某种形式的注入风险测试，发生安全事件的最大率为 19%，平均率为 3.37%，匹配到此类别的 33 个 CWE 共发生 27.4 万次，是出现第二多的风险类别。原“A07:2017-跨站脚本（XSS）”在 2021 年版中被纳入此风险类别。
A04	不安全设计 (Insecure Design)	2021 年版的一个新类别，其重点关注与设计缺陷相关的风险。如果我们真的想让整个行业“安全左移” ，我们需要更多的威胁建模、安全设计模式和原则，以及参考架构。不安全设计是无法通过完美的编码来修复的；因为根据定义，所需的安全控制从来没有被创建出来以抵御特定的安全攻击。
A05	安全配置错误 (Security Misconfiguration)	排名上升一位。90%的应用程序都进行了某种形式的配置错误测试，平均发生率为 4.5%，超过 20.8 万次的 CWE 匹配到此风险类别。随着可高度配置的软件越来越多，这一类别的风险也开始上升。原“A04:2017-XML External Entities(XXE) XML 外部实体”在 2021 年版中被纳入此风险类别。
A06	自带缺陷和过时的组件(Vulnerable and Outdated Components)	排名上升三位。在社区调查中排名第 2。同时，通过数据分析也有足够的数据进入前 10 名，是我们难以测试和评估风险的已知问题。它是唯一一个没有发生 CVE 漏洞的风险类别。因此，默认此类别的利用和影响权重值为 5.0。原类别命名为“A09:2017-Using Components with Known Vulnerabilities 使用含有已知漏洞的组件”。
A07	身份识别和身份验证错误 (Identification and Authentication Failures)	排名下滑五位。原标题“A02:2017-Broken Authentication 失效的身份认证”。现在包括了更多与识别错误相关的 CWE。这个类别仍然是 Top 10 的组成部分，但随着标准化框架使用的增加，此类风险有减少的趋势。
A08	软件和数据完整性故障(Software and Data Integrity	2021 年版的一个新类别，其重点是：在没有验证完整性的情况下做出与软件更新、关键数据和 CI/CD 管道相关的假设。此类别共有 10 个匹配的 CWE 类别，并且拥有最高的平均加权影响值。原

	Failures)	“A08:2017-Insecure Deserialization 不安全的反序列化”现在是本大类的一部分。
A09	安全日志和监控故障(Security Logging and Monitoring Failures)	排名上升一位。来源于社区调查（排名第 3）。原名为“A10:2017-Insufficient Logging & Monitoring 不足的日志记录和监控”。此类别现扩大范围，包括了更多类型的、难以测试的故障。此类别在 CVE/CVSS 数据中没有得到很好的体现。但是，此类故障会直接影响可见性、事件告警和取证。
A10	服务端请求伪造(Server-Side Request Forgery)	2021 年版的一个新类别，来源于社区调查（排名第 1）。数据显示发生率相对较低，测试覆盖率高于平均水平，并且利用和影响潜力的评级高于平均水平。加入此类别风险是说明：即使目前通过数据没有体现，但是安全社区成员告诉我们，这也是一个很重要的风险。

通过上面的 TOP 10 的漏洞威胁， 如果不是从事安全工作或者不了解安全信息的同学来说，理解起来有点费劲，但是，我举个例子，你就会知道：

• 弱密码

还记得前段时间，某两过的战争中，其中一国的国防系统登录账号及密码曝出为“admin”和“123456”。

这是什么概念呢？

我举个例子：虽然你每天都会锁门，但是，你的钥匙，就放在门边上。

你说，你这门锁的有什么意义呢？

当然，如果对瞎子来说，可能有点费劲，毕竟钥匙放到哪里，瞎子是看不到的，但是又有哪个瞎子会去陌生人家里 “串门”呢？

2.3 安全测试目的

到这里，既然知道信息安全的重要性，以及信息威胁的种类，我们就需要避免这些漏洞就好了。

但是，如何避免呢？又如何能确定自己开发的模块就没有安全漏洞呢？

这个时候，就需要安全测试的介入了。

在安全测试介入前，我们需要了解安全测试的目的，只有知道其目的，才能更好的进行测试。

我也用一段话来回答：

安全测试的目的，就是查找软件自身程序设计中存在的安全隐患，并检查应用程序对非法侵入的防范能力， 根据安全指标不同测试策略也不同，如果遵循相同的原则，去证明软件的安全性，将有利于软件安全测试的工作规范的进行，有利于软件安全测试工作的发展。

俗话说，要想彻底打败”敌人”，除了自己有高超的武艺外，还需要有“武器”的加持。

所以，在了解安全测试目的后，我我们就进入安全测试的下一个阶段，即：学习掌握安全测试工具。

3、安全测试工具介绍

关于移动 APP 的安全测试工具，有很多，这里，我只介绍 3 款，即：

-Mobsf;

-QARK

-Drozer

我想，做过移动 APP 安全渗透测试的同学，对这三款工具，应该都不陌生了，

这三款，应该在移动 APP 的安全测试界，算得上是主流的了。

这里，我也对这三款进行简单介绍，希望你能对它们有一个初步的认识。

3.1 MobSF

定义：

Mobsf，全称为 Mobile Security Framework， 是一款自动化移动 App 安全测试工具，适用于 iOS 和 Android，可执行动态、静态分析和 Web API 测试。

功能：

Mobsf 可用于对 Android 和 iOS 应用进行快速安全分析。

支持 binaries（IPA 和 APK）以及 zipped 的源代码。

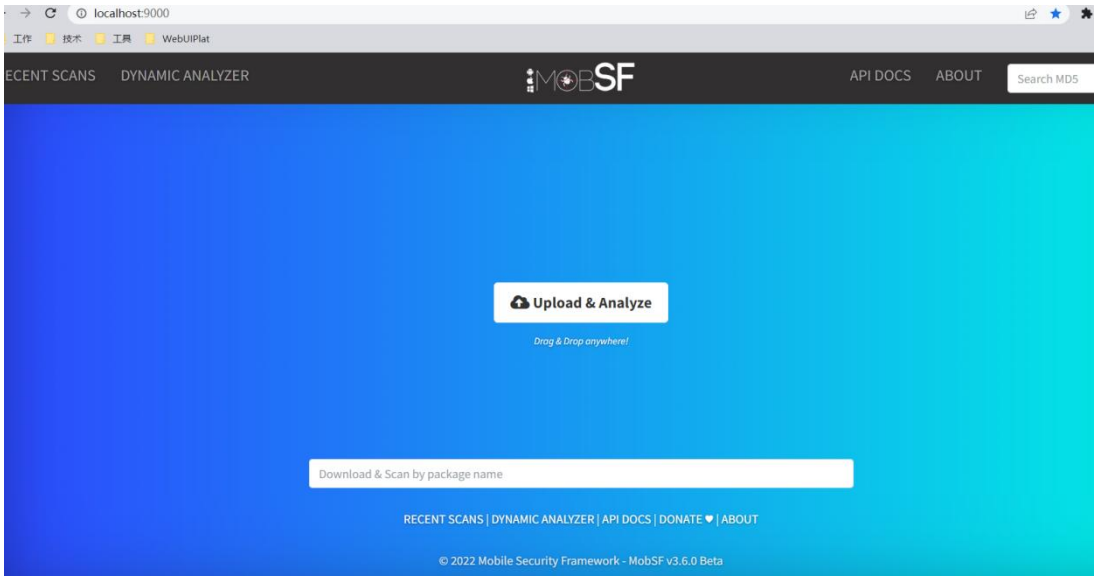
特点：

一款开源移动 APP 安全测试工具；

可以在本地环境托管，不用担心信息泄露；

对 Android 、IOS、Windows 三个平台的移动 APP 进行安全性分析。

展示图：



3.2 QARK

定义：

QARK 全称 Quick Android Review Kit，是有领英(LinkedIn)开发，是一款静态代码分析工具，可以快速 Android 审查工具包，这个工具可用来检查 Android 应用的源代码和打包的 APK 中常见的安全漏洞。

特点：

一款开源移动 APP 安全测试工具；

能提供详细的漏洞报告；

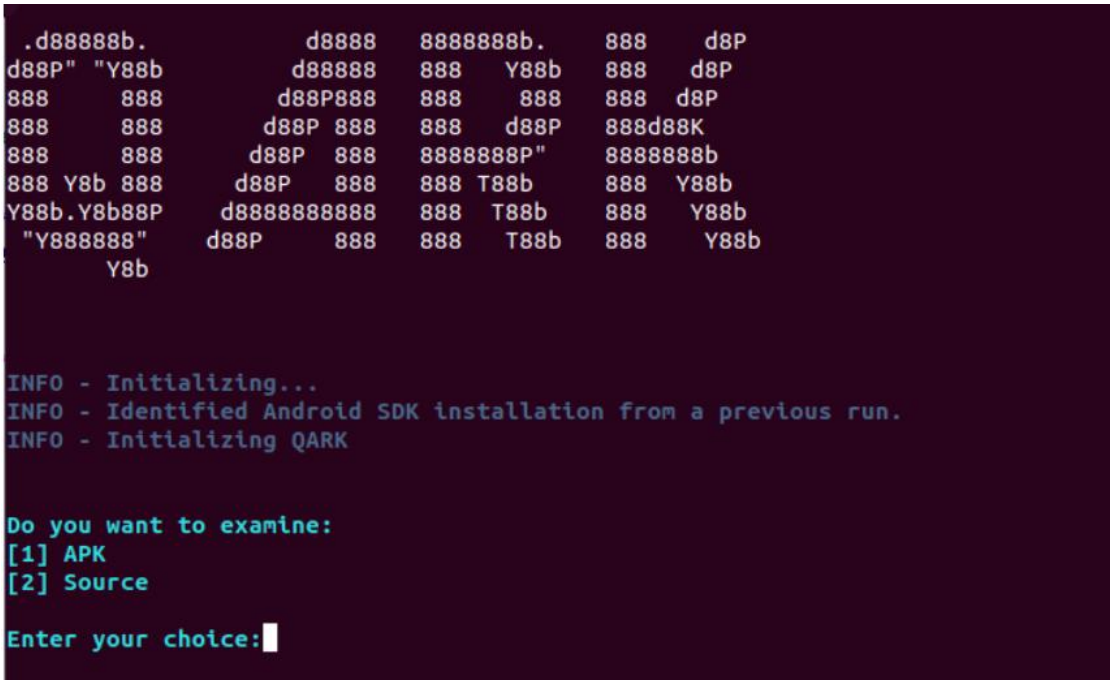
能扫描移动 App 中的所有元素，查找安全威胁；

它可以生成 ADB 命令，甚至是功能齐全的 APK，从而将假设的漏洞转化为有效的“POC” 漏洞利用；

弊端：

QARK 会将 Android 应用程序反编译回原始源代码，这在某些情况下可能属于违法行为，谨慎使用。

展示图：



3.3 Drozer

定义：

Drozer 是由 MWR InfoSecurity 开发的 App 安全测试框架，分为两部分，即：

①安装在 PC 端的控制台；

② 安装在终端上的代理 APP；

交互：

可以通过与 dalivik VM 交互、与其他应用程序的 IPC 端点交互、与底层操作系统的交互，来避免正处于开发阶段或者正处于部署于组织的 Android 应用程序和设备暴露出安全风险；

特点：

一款开源移动 APP 安全测试工具；

可以基于真机和模拟器进行测试，不需要 USB 调试或其他开发工具即可使用；

可以进行自动扩展，进行 app 的安全测试；

弊端：

只支持 python2.x 的版本，不持之 python3.x 的版本；

PC 端与移动端交互时，需要 adb 命令；

展示图：

PC 端

移动端

我之所以列举这三款工具，

第一，这三款工具在移动 APP 的安全测试中还算比较主流，并且使用的人数也挺可观；

第二，因为这三款工具，有的可以直接分析静态代码和动态交互，有的可以直接进行在移动 APP 上进行操作；

第三，这三款工具，测试报告给我的感觉非常的完善；

第四，这三款工具，我使用的还算蛮久的…

今天我主要介绍 Mobsf 这款工具，至于其他两款，后续有时间会逐一进行介绍。

4、环境搭建与服务部署

4.1 环境准备

前面提到，Mobsf 可以部署到本地环境，这就避免数据的与云交互，毕竟，“家丑不可外扬”。

同样，这里我以 Windows 为例，需要安装的工具，如下：

-Python 3.9

-Windows11_x64

-Jdk 1.8

-Microsoft Visual C++ Build Tools

-OpenSSL

如果你想部署在 Linux 系统或者安装在 Mac 上，这些工具也是需要准备的。

4.2 工具下载

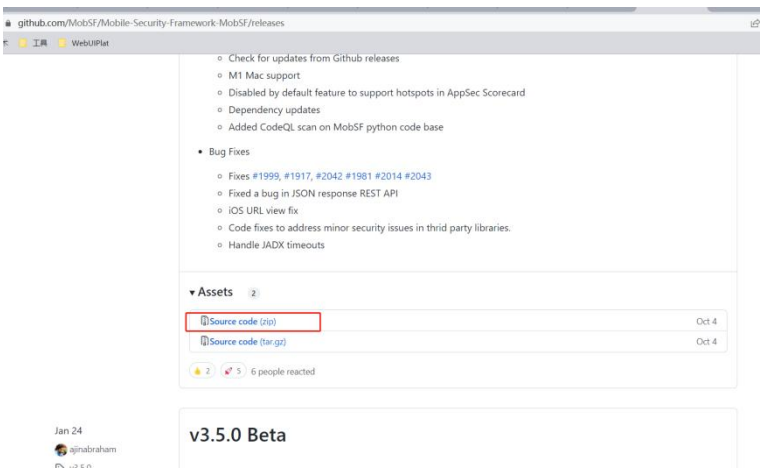
因为 Mobsf 的安装方式有两种，即：docker 安装 和 下载源码安装。

考虑到大部分同学，不太喜欢安装 docker，这里，我就以下载源码方式安装。

工具下载：

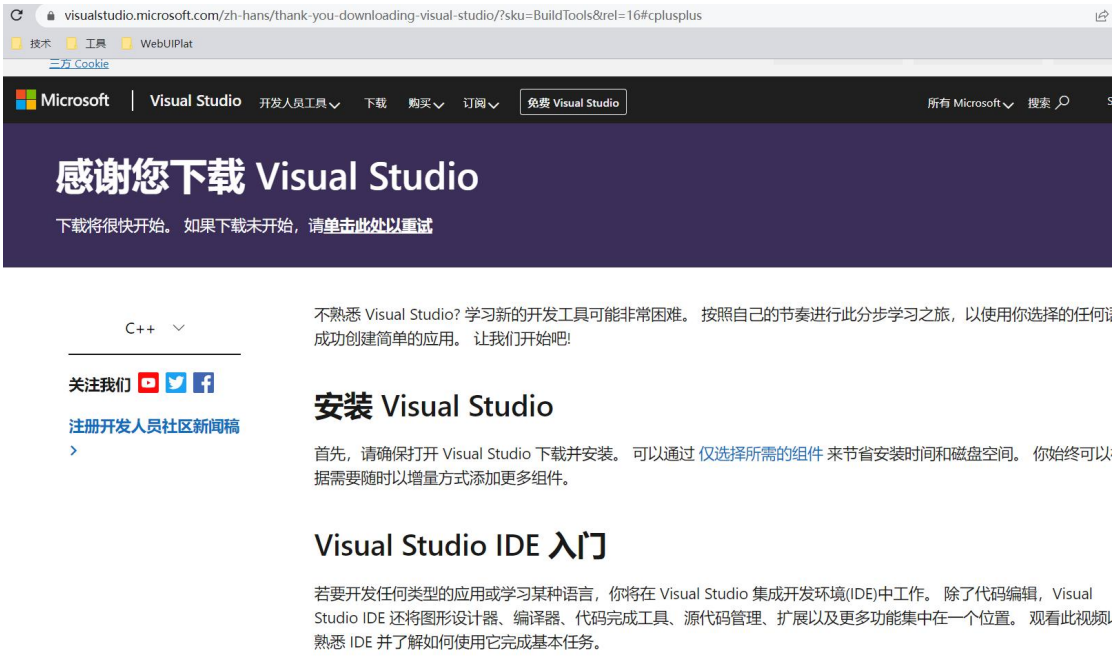
-Mobsf 源码下载地址：

https://github.com/MobSF/Mobile-Security-Framework-MobSF/releases

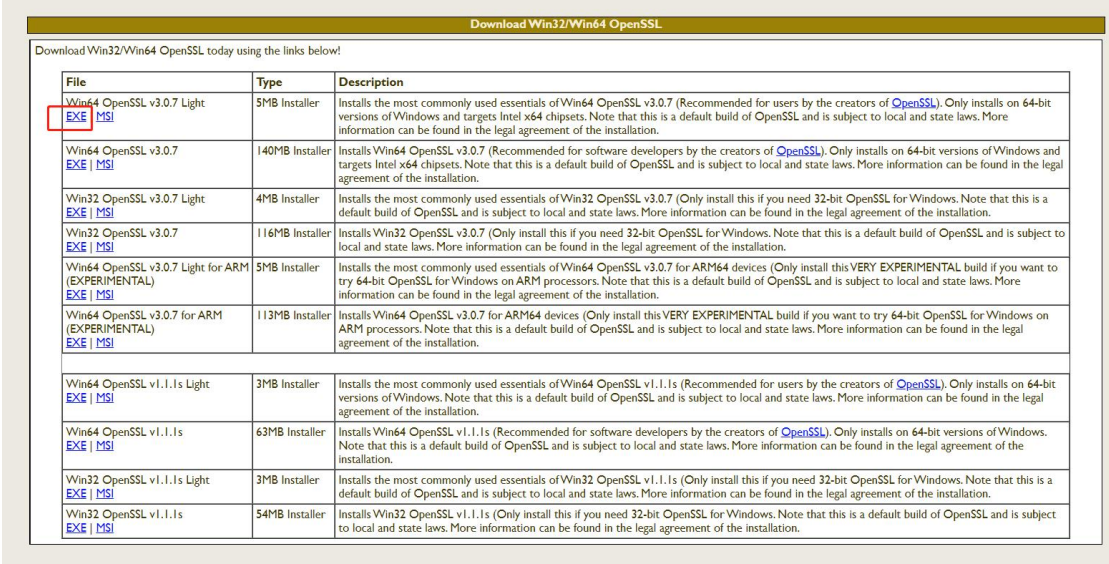


选择 Mobsf 的 V3.6.0 的版本的 Source Code .zip 包进行下载下载

-Microsoft Visual C++ Build Tools 下载地址：
https://visualstudio.microsoft.com/zh-hans/thank-you-downloading-visual-studio/?sku=BuildTools&rel=16

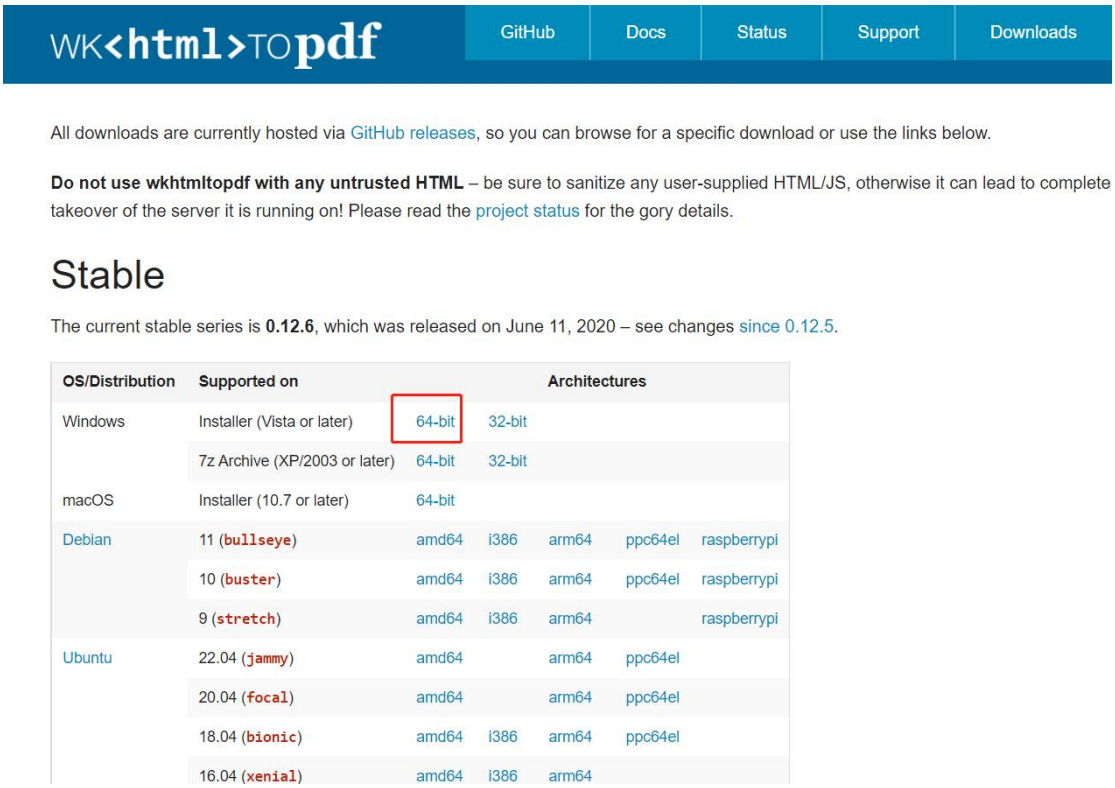


-OpenSSL 下载地址：http://slproweb.com/products/Win32OpenSSL.html



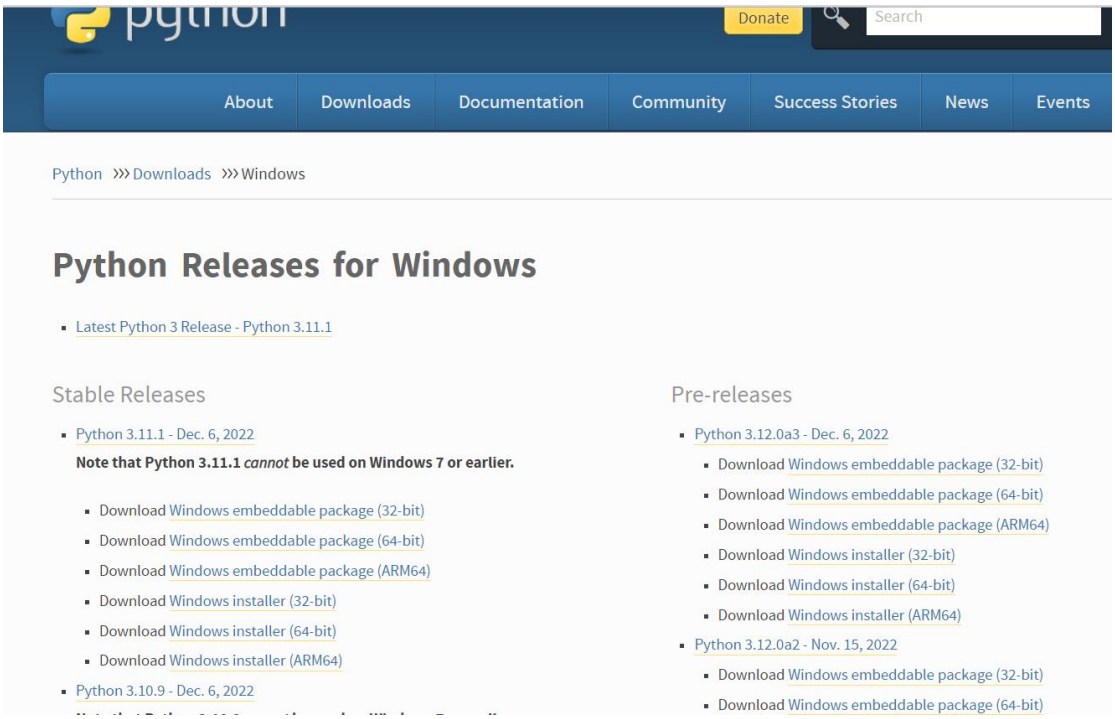
根据系统，选择你需要的版本，我下载的是 win64 的 v3.0.7 EXE 版本

-Wkhtmltopdf 下载地址：https://wkhtmltopdf.org/downloads.html



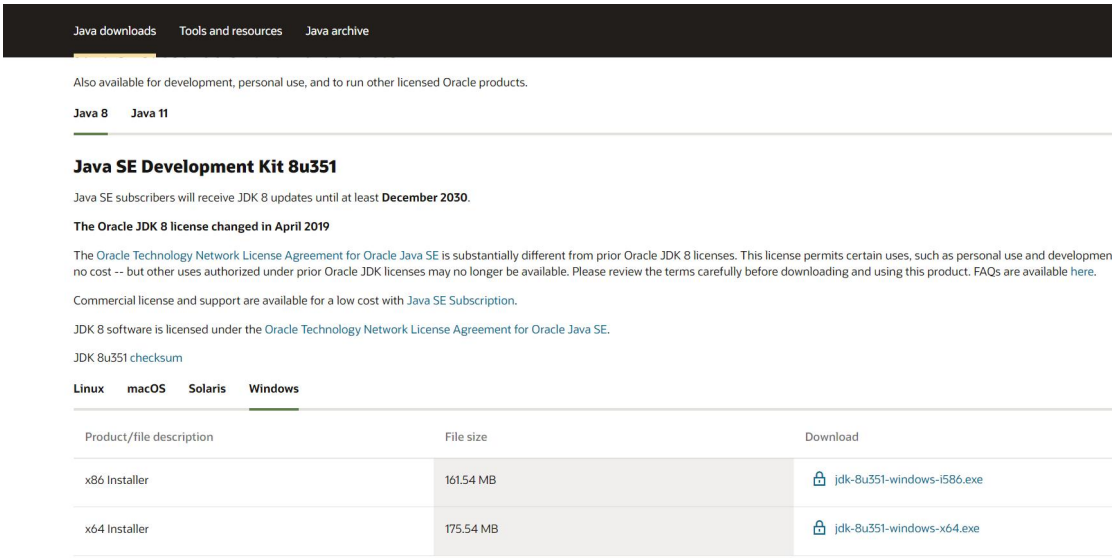
同样，根据 PC 的系统，你选择下载的版本，我下载的是 Windows Installer 64-bit 版本：

- Python 下载地址：https://www.python.org/downloads/



因为 Mobsf 支持 python2.x 和 python3.x，所以，你可以放心，大胆的下载 Python3.11 的版本。

- jdk 下载地址： https://www.oracle.com/java/technologies/downloads/



这里说明一下：

1、如果你有 git，那么你可以直接执行如下命令：

git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git;

2、Wkhtmltopdf 是可以把 Mobsf 的测试报告以 PDF 文件格式展示及下载；

3、openSSL 与 Visual C++ Build Tools 是必须要下载的，否则 Mobsf 无法安装；

4、Linux/Mac 安装命令：

Git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git

cd Mobile-Security-Framework-MobSF

./setup.sh

4.3 服务安装配置

上面的内容都下载完成后，就需要安装了。

-先安装：jdk、python、openSSL、Visual C++ Build Tools

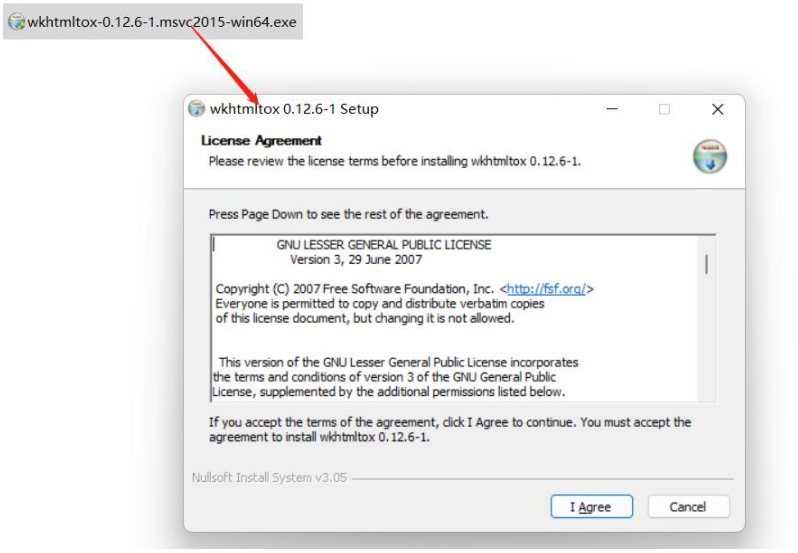
-再安装：Mobsf、Wkhtmltopdf

都是傻瓜式安装，前面的 jdk、python 就不多说了，这里主要说 Wkhtmltopdf 与 Mobsf 的安装步骤。

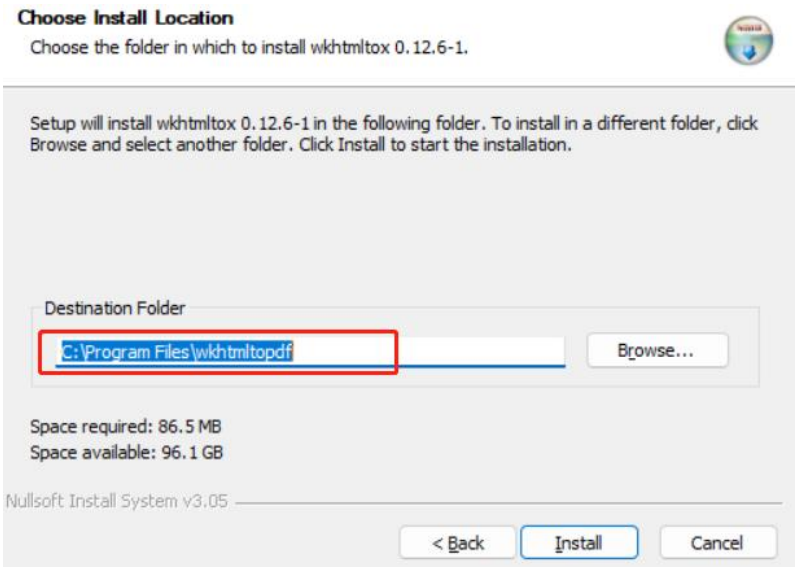
4.3.1 Wkhtmltopdf 安装与配置

1、安装

安装方式也很简单，直接双击安装即可。

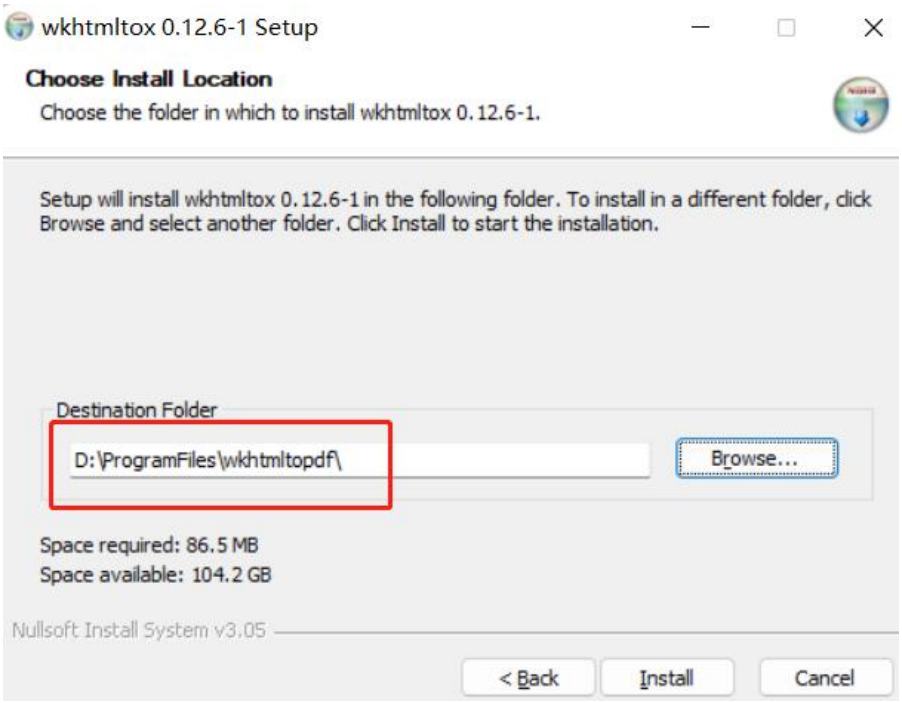


点击 I Agree 按钮，进入到选择安装文件页面，如下：



默认安装路径为：C:\Program Files\wkhtmltopdf

这里一定要更改默认安装路径，因为 wkhtmltopdf 的安装路径文件夹名称不能有空格，这一点，一定要记住。



这里，我把安装路径更改为 D:\ProgramFiles\wkhtmltopdf。

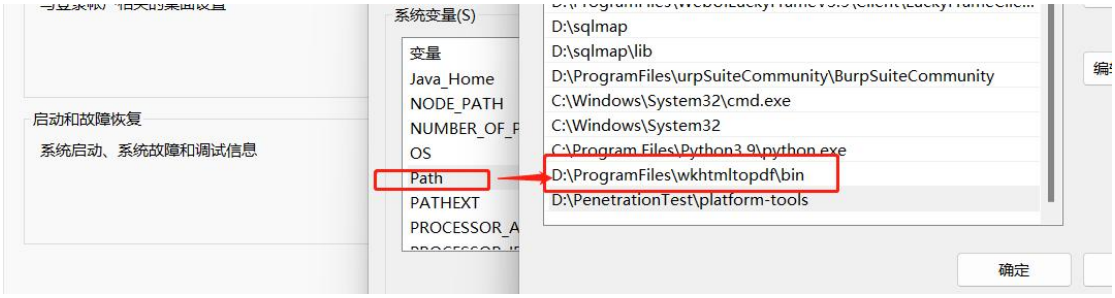
然后点击 Install 就等待这安装完成就可以。

2、配置环境变量

wkhtmltopdf 需要添加环境变量，否则无法启动 pdf 文件转换功能。

把\wkhtmltopdf\bin 添加到 path 中即可，可以参考下图。

\wkhtmltopdf\bin 路径



添加到 Path 中

3、应用程序解读

\wkhtmltopdf\bin 文件夹下有两个.exe 执行程序

-wkhtmltoimage.exe：生成图片；

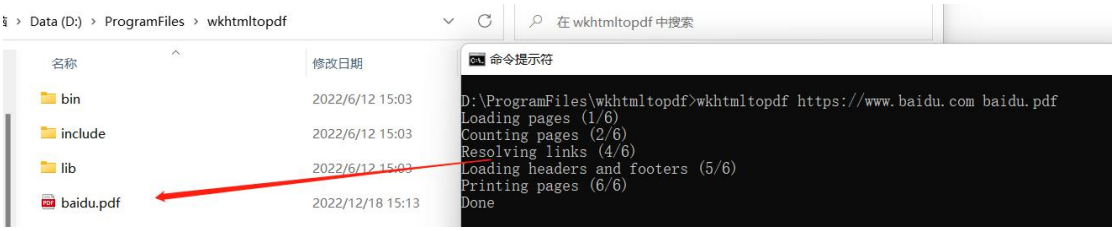
-wkhtmltopdf.exe：生成 pdf；

4、验证

①验证生成 PDF 文件

在任意目录下，输入命令：wkhtmltopdf https://www.baidu.com baidu.pdf

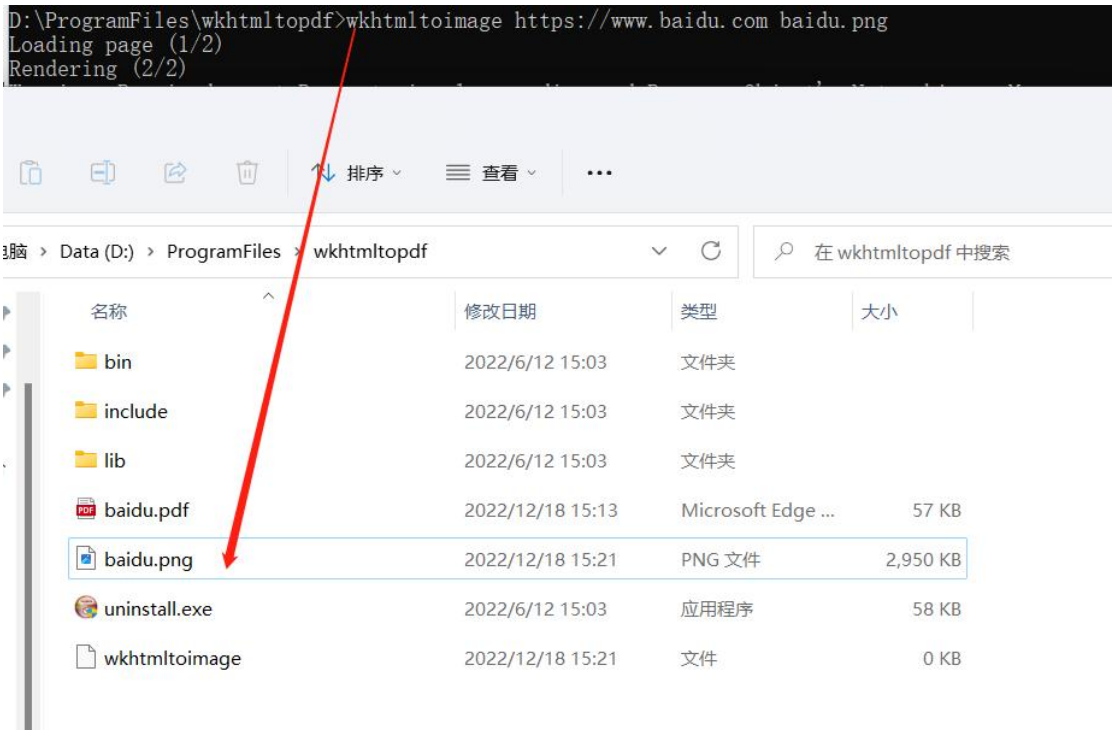
这里为了看的更直观，我就在 wkhtmltopdf 根目录下，执行命令。



生成 PDF 文件，如下：

②验证生成图片文件

同样，输入命令：wkhtmltoimage https://www.baidu.com baidu.png



生成图片，如下：

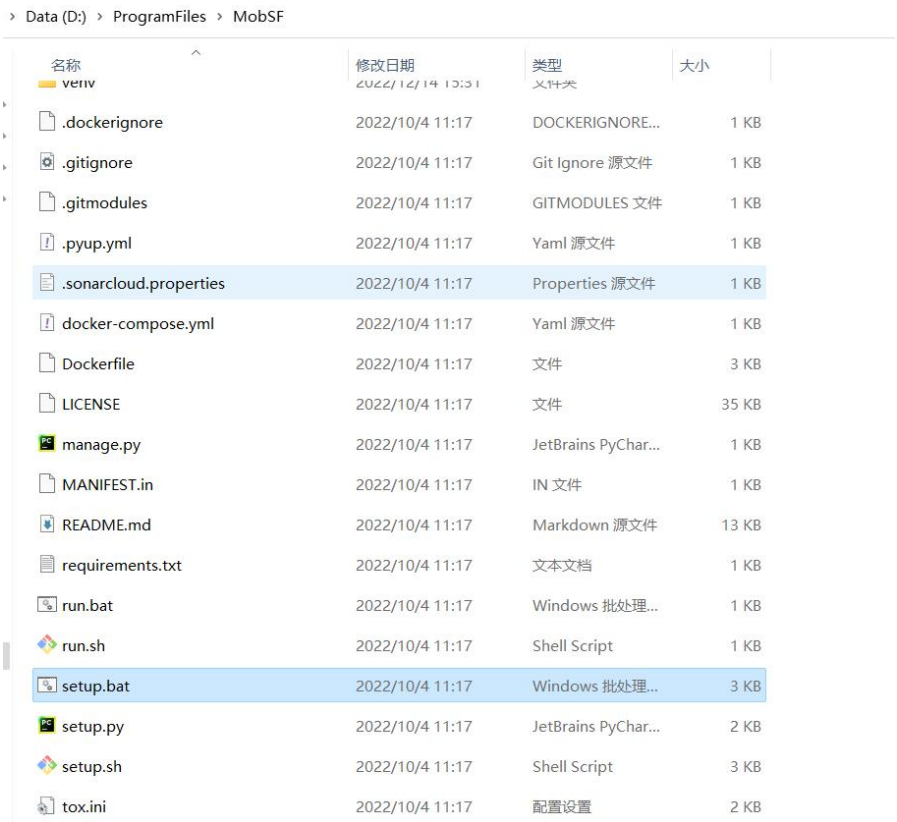
到这里， 说明 Wkhtmltopdf 安装成功。

4.3.2Mobsf 安装与配置

1、把下载的 Mobsf.zip 进行解压。

这里我解压到 D 盘的 MobSF 文件夹下。

路径不要有中文，不要有空格;



2、安装依赖，这里有两种方式：

①双击文件安装：在 MobSF 文件夹下，双击 setup.bat ，就会进行自动安装所需要的第三方库；

②使用命令安装：pip install -r requirements.txt；

这里，我也整理出需要安装的第三方依赖库，如下图：

```
Django>=3.1.5
lxml>=4.6.2
rsa>=4.7
biplist>=1.0.3
requests>=2.25.1
bs4>=0.0.1
colorlog>=4.7.2
macholib>=1.14
whitenoise>=5.2.0
waitress>=1.4.4;platform_system=='Windows'
gunicorn>=20.0.4;platform_system!='Windows'
psutil>=5.8.0
shelljob>=0.6.2
asn1crypto>=1.4.0
oscrypto>=1.2.1
distro>=1.5.0
IP2Location==8.8.1
lief>=0.12.1
http-tools>=2.1.0
libsast>=1.5.1
pdfkit>=0.6.1
google-play-scraper>=0.1.2
androguard==3.4.0a1
apkid==2.1.4
quark-engine==22.6.1
frida==15.2.2
tldextract==3.3.1
openstep-parser==1.5.3
# For semgrep & mitmproxy
ruamel.yaml==0.16.13 # pyup: ignore
click==8.0.1 # pyup: ignore
decorator==4.4.2 # pyup: ignore
```

这么多第三方库需要安装，你可以利用这段时间，去打杯咖啡，或者泡杯茶，是泡茶，可别泡别的哦。

如果你不愿意动弹，那你可以利用这段时间，来思考这样一个问题：

你学习安全渗透测试，为了什么？

因为在软件测试领域中，安全渗透测试与性能测试，相对于自动化测试(UI、API、APP)来说，是需要相当深厚的功底。

我之所以问这个问题，我希望你会对你的职业生涯有一个规划，而不是随波逐流。

这里强调一下：

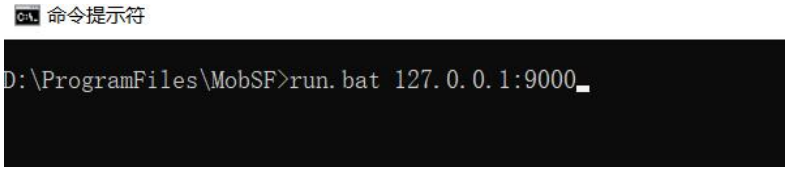
新版的 MobSF 安装完成后，不需要手动执行初始数据库的操作，直接启动 Mobsf 就可以。

4.4 启动服务

在经过一段时间的等待，Mobsf 终于安装完成。

接着，我们就去验证，Mobsf 是否可以正常启动。

在 Mobsf 的根目录下输入命令：run.bat 127.0.0.1:9000

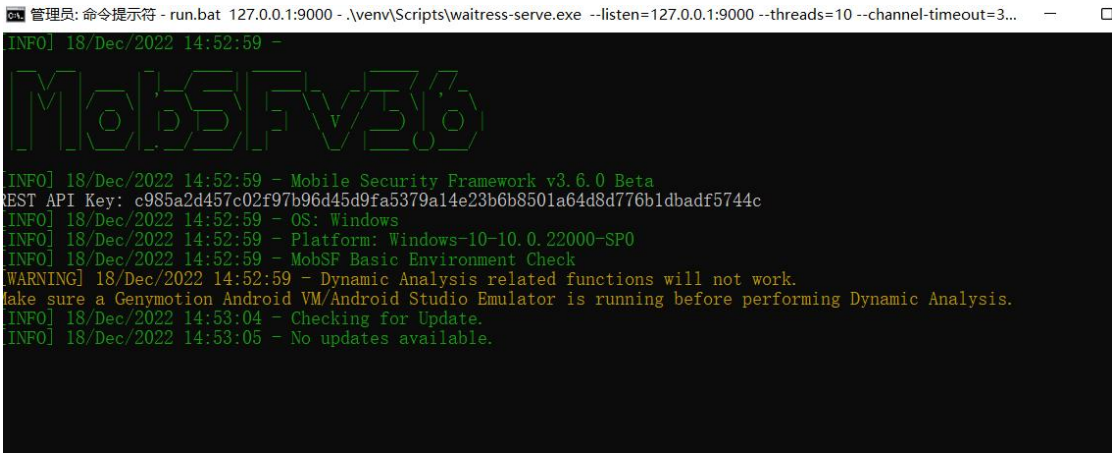


这里需要注意两点：

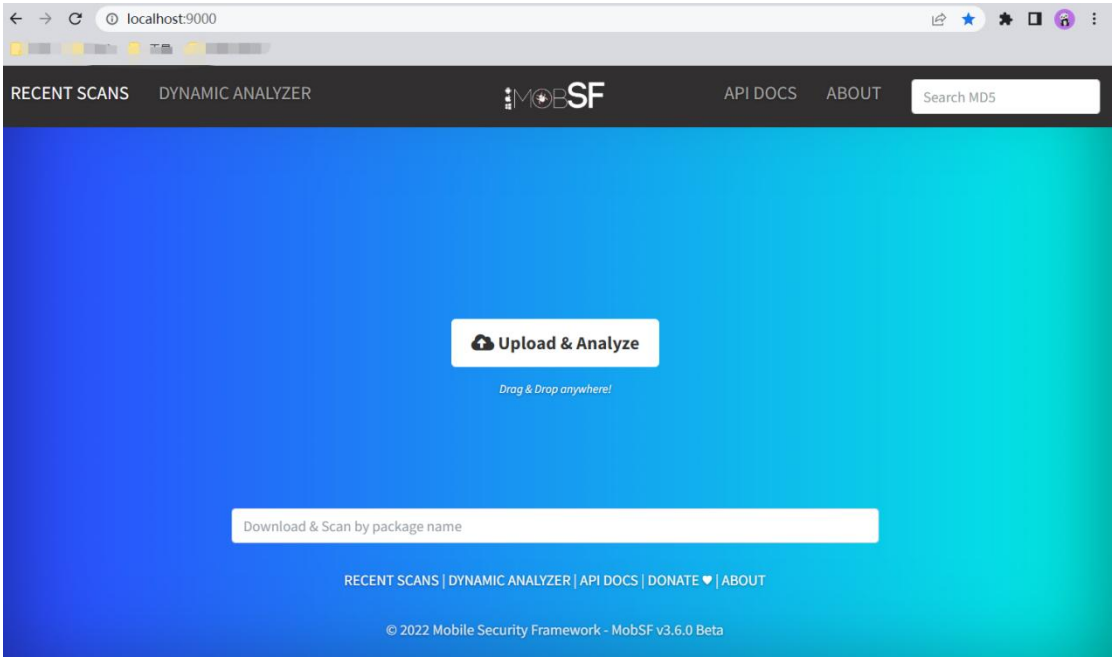
①Run.bat 后面必须加上 服务地址 和端口号；

②端口号不能被别的服务占用，否则无法启动。

经过不懈的努力， 当你看到 MobSFv3.6 的笑脸，说明服务启动了。



接着，在浏览器中输入 ：127.0.0.1:9000 (或 localhost:9000)，看到下图的界面，说明 Mobsf 平台已经成功搭建完成。



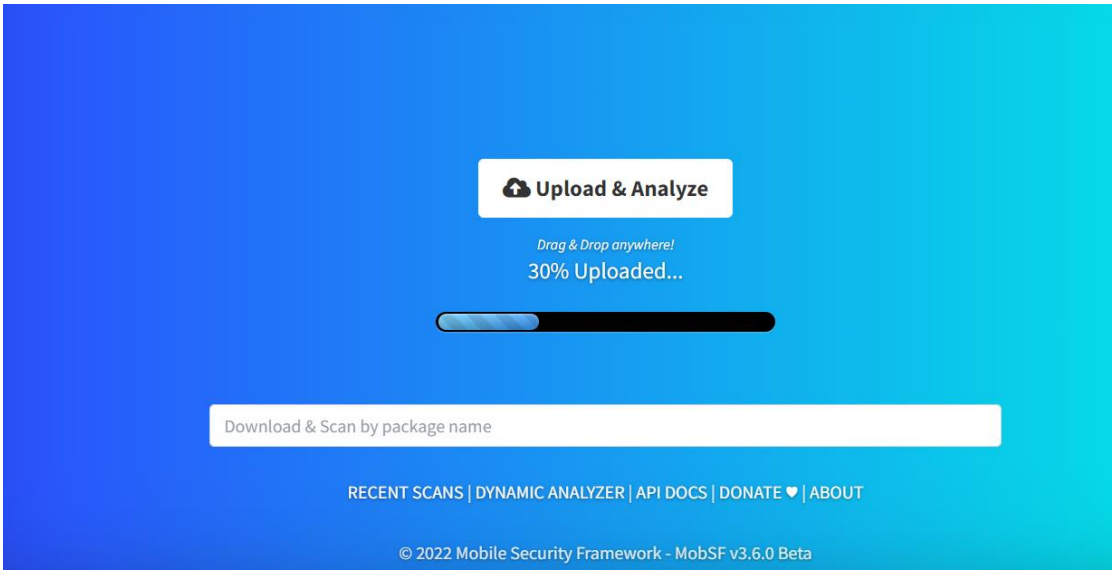
此时，你的小心脏是不是有点激动，我们终于可以进行 apk 扫描了，里最后的成功，只差一步了。

我们接着往前走，挥一挥衣袖，不遗留一个漏洞。

5、Mobsf 使用

5.1 包上传

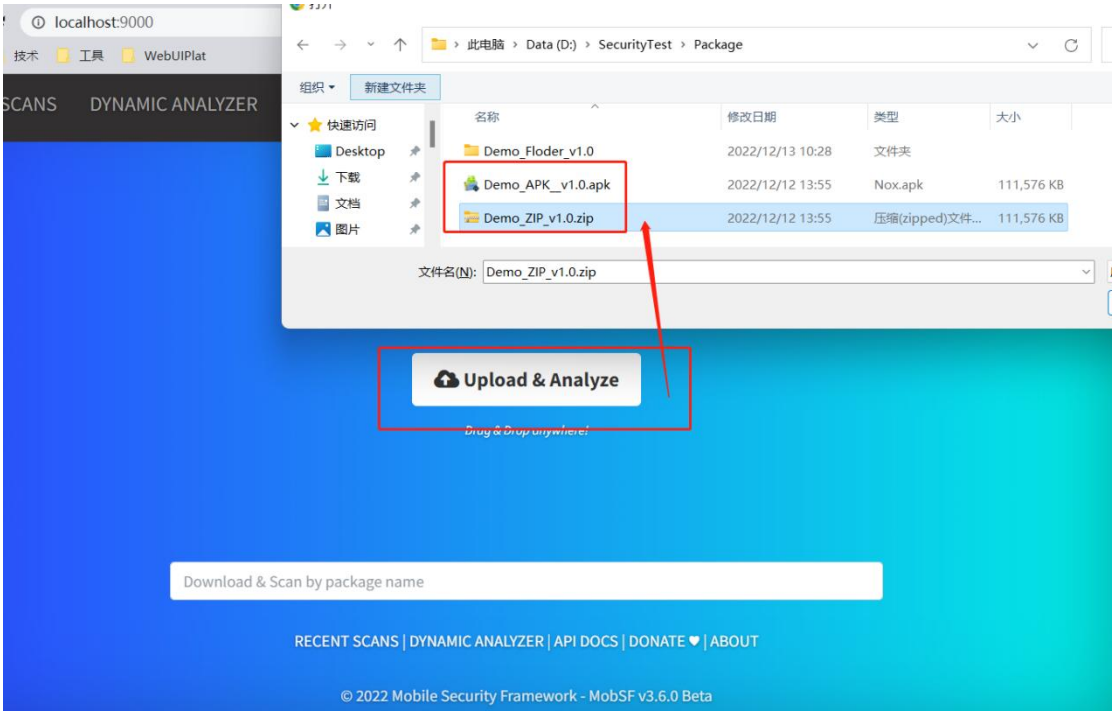
1、点击 Upload& Analyze 按钮，选择 Demo_AKP_v1.0.apk



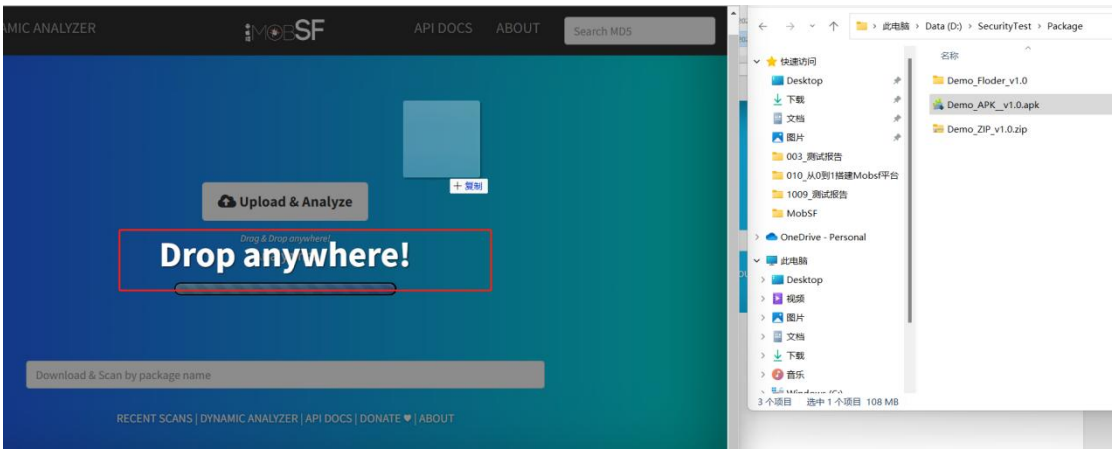
上传完成，就等待这 Mobsf 自动分析代码的漏洞。

注：

① 上传方式，支持 Drop 和 upload 两种方式

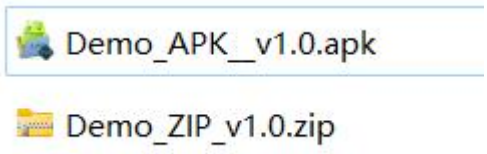


Upload 方式



Drop 方式

③ 上传包的格式，支持 .apk 和 .zip 两种:



分析过程中，时间会有一点点的长，此时，一定要经得住”孤独”。

5.2Mobsf 静态分析原理

5.2.1 分析原理流程图

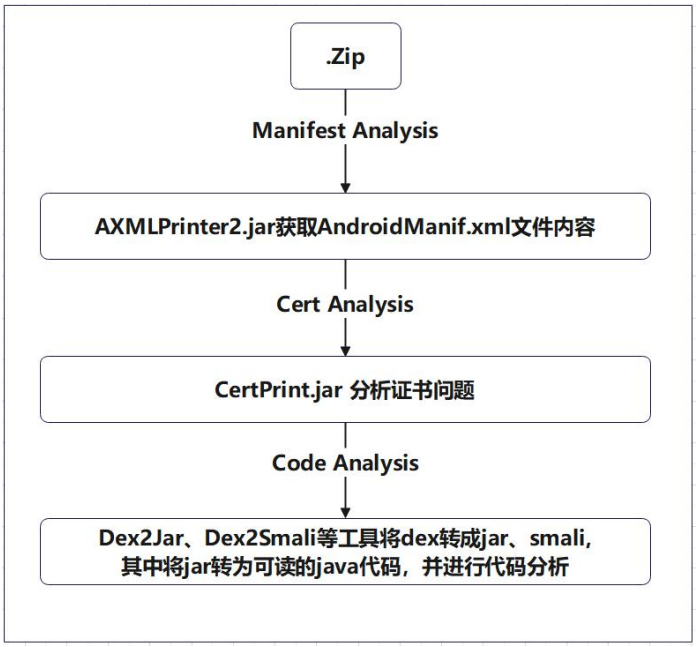
Mobsf 的静态分析主要包含三部分：Manifest Analysis、Cert Analysis、Code Analysis，如下图：

master · Mobile-Security-Framework-MobSF / mobsf / StaticAnalyzer / views / android / — Go to file

superpoussin22 Apktool 2.7.0 update (#2082) ... — 1016a3d 4 days ago — History

..		
rules	HOTFIX: libsast upgrade and rule upgrade	10 months ago
__init__.py	HOTFIX: setup.py and directory refactoring	2 years ago
android_manifest_desc.py	[EFR-02]Enterprise Feature Request - False Positive Triaging (#2000)	2 months ago
binary_analysis.py	Remove RELRO (#1978)	6 months ago
cert_analysis.py	Update cert_analysis.py (#1948)	6 months ago
code_analysis.py	Multiple Features on v3.5.0 (#1881)	last year
converter.py	[EFR-02]Enterprise Feature Request - False Positive Triaging (#2000)	2 months ago
db_interaction.py	[EFR-02]Enterprise Feature Request - False Positive Triaging (#2000)	2 months ago
dvm_permissions.py	TLS/SSL Security Tester (#1726)	last year
find.py	Multiple Features on v3.5.0 (#1881)	last year
generate_downloads.py	Feature Parity Allow iOS IPA download (#1977)	6 months ago
icon_analysis.py	Android APK support extracting icon SVG from XML (#2060)	last month
manifest_analysis.py	Apktool 2.7.0 update (#2082)	4 days ago
manifest_view.py	HOTFIX: setup.py and directory refactoring	2 years ago
network_security.py	Multiple Features on v3.5.0 (#1881)	last year

这里，为了便于你理解，我整理成流程图，如下：



5.2.2 Manifest Analysis 详解

Manifest Analysis 主要功能：

①对 AndroidManifest.xml 进行分析；

②提取 permission、granturipermissions、application、activties、services、intents 等，分析所有权限，并且对权限进行 “正常”、”危险”、”签名”、”系统”分级；

③对 allowBackup、debuggable、exported 等属性配置进行检测；

这里，我也把代码摘出来，便于理解。

```
# PERMISSION
        for permission in permissions:
            if permission.getAttribute('android:protectionLevel'):
                protectionlevel = permission.getAttribute(
                    'android:protectionLevel')
                if protectionlevel == '0x00000000':
                    protectionlevel = 'normal'
                elif protectionlevel == '0x00000001':
                    protectionlevel = 'dangerous'
```

```
                elif protectionlevel == '0x00000002':
                    protectionlevel = 'signature'
                elif protectionlevel == '0x00000003':
                    protectionlevel = 'signatureOrSystem'

                permission_dict[permission.getAttribute(
                    'android:name')] = protectionlevel
            elif permission.getAttribute('android:name'):
                permission_dict[permission.getAttribute(
                    'android:name')] = 'normal'

        # APPLICATIONS
        for application in applications:
            # Esteve 23.07.2016 - begin - identify permission at the
            # application level
            if application.getAttribute('android:permission'):
                perm_appl_level_exists = True
                perm_appl_level = application.getAttribute(
                    'android:permission')
            else:
                perm_appl_level_exists = False
            # End
            if application.getAttribute('android:usesCleartextTraffic') == 'true':
                ret_list.append(('clear_text_traffic', (), ()))
            if application.getAttribute('android:directBootAware') == 'true':
                ret_list.append(('direct_boot_aware', (), ()))
            if application.getAttribute('android:networkSecurityConfig'):
                item = application.getAttribute('android:networkSecurityConfig')
                ret_list.append(('has_network_security', (item,), ()))
                do_netsec = item
            if application.getAttribute('android:debuggable') == 'true':
                ret_list.append(('app_is_debuggable', (), ()))
                debuggable = True
```

```
        if application.getAttribute('android:allowBackup') == 'true':
            ret_list.append(('app_allowbackup', (), ()))
        elif application.getAttribute('android:allowBackup') == 'false':
            pass
        else:
            ret_list.append(('allowbackup_not_set', (), ()))
        if application.getAttribute('android:testOnly') == 'true':
            ret_list.append(('app_in_test_mode', (), ()))
```

5.2.3 Code Analysis 详解

Code Analysis 功能：

① 使用 Dex2Jar 将 dex 转变为 jar 文件；

②使用 Dex2Smali 将 dex 转变为 smali 代码；

在 code_analysis.py 文件中，包含对 Code 和 API 的分析、NIAP 扫描、邮箱和提取路径等。

1、Code 和 API 的分析，代码：

```
# Code and API Analysis
    code_findings = scan(
        code_rules.as_posix(),
        {'.java', '.kt'},
        [src],
        skp)
    api_findings = scan(
        api_rules.as_posix(),
        {'.java', '.kt'},
        [src],
        skp)
```

2、NIAP 扫描，代码如下：

```
# NIAP Scan
        logger.info('Running NIAP Analyzer')
        niap_findings = niap_scan(
            niap_rules.as_posix(),
            {'.java', '.xml'},
            [src],
            manifest_file,
            None)
```

NIAP 扫描 .java 和.xml 格式文件内容。

3、Extraxt URLs 和 Emails 代码，如下：

```
# Extract URLs and Emails
        for pfile in Path(src).rglob('*'):
            if (
                (pfile.suffix in ('.java', '.kt')
                    and any(skip_path in pfile.as_posix()
                            for skip_path in skp) is False)
            ):
                content = None
                try:
                    content = pfile.read_text('utf-8', 'ignore')
                    # Certain file path cannot be read in windows
                except Exception:
                    continue
                relative_java_path = pfile.as_posix().replace(src, '')
                urls, urls_nf, emails_nf = url_n_email_extract(
                    content, relative_java_path)
                url_list.extend(urls)
                url_n_file.extend(urls_nf)
```

```
                email_n_file.extend(emails_nf)
```

代码没有任何难度，就是提取 Email 和 url 信息。

5.3.4 Cert Analysis 详解

工作流程：

先尝试获取 Hardcoded Certificates/Keystores，再通过 CertPrint.jar 解析 apk 中的证书信息，完成证书相关的问题分析。

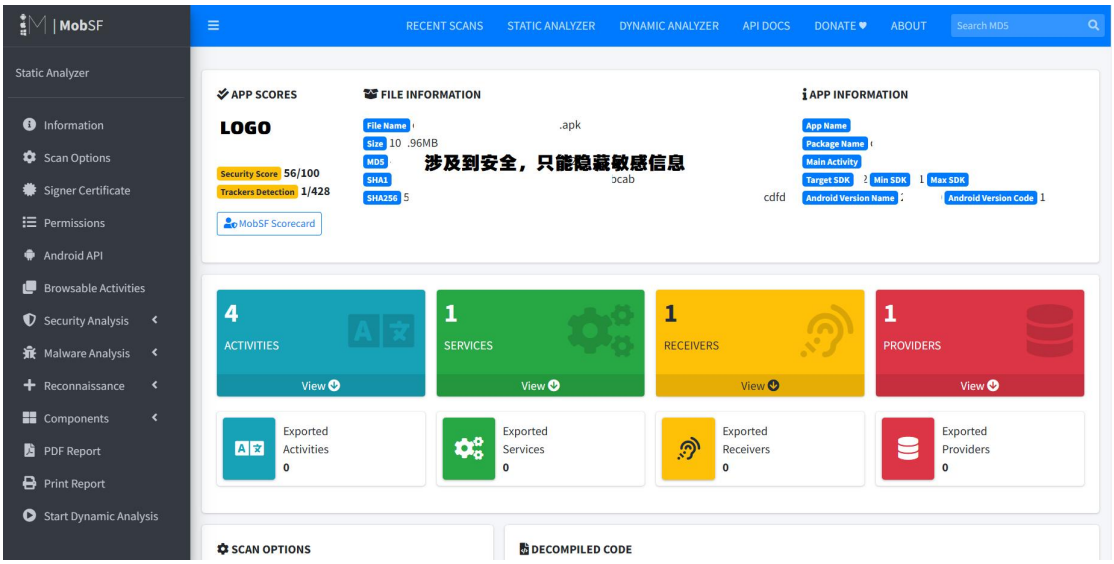
我也截取一部分代码，如下：

```
def get_hardcoded_cert_keystore(files):
    """Returns the hardcoded certificate keystore."""
    try:
        logger.info('Getting Hardcoded Certificates/Keystores')
        findings = []
        certz = []
        key_store = []
        for file_name in files:
            if '.' not in file_name:
                continue
            ext = file_name.split('.')[-1]
            if re.search('cer|pem|cert|crt|pub|key|pfx|p12|der', ext):
                certz.append(escape(file_name))
            if re.search('jks|bks', ext):
                key_store.append(escape(file_name))
        if certz:
            desc = 'Certificate/Key files hardcoded inside the app.'
            findings.append({'finding': desc, 'files': certz})
        if key_store:
            desc = 'Hardcoded Keystore found.'
            findings.append({'finding': desc, 'files': key_store})
        return findings
```

```
        except Exception:
            logger.exception('Getting Hardcoded Certificates/Keystores')
    def cert_info(app_dir, app_file):
        """Return certificate information."""
        try:
            logger.info('Reading Code Signing Certificate')
            manifestfile = None
            manidat = ''
            cert_info = ''
            certlist = []
            cert_path = os.path.join(app_dir, 'META-INF/')
            apk_file = os.path.join(app_dir, app_file)
            hashfunctions = {
                'md5': hashlib.md5,
                'sha1': hashlib.sha1,
                'sha256': hashlib.sha256,
                'sha512': hashlib.sha512,
            }
            files = [f for f in os.listdir(
                cert_path) if os.path.isfile(os.path.join(cert_path, f))]
            a = APK(apk_file)
            if a.is_signed():
                certlist.append('APK is signed')
            else:
                certlist.append('Missing certificate')
            certlist.append('v1 signature: {}'.format(a.is_signed_v1()))
            certlist.append('v2 signature: {}'.format(a.is_signed_v2()))
            certlist.append('v3 signature: {}'.format(a.is_signed_v3()))
            certs = set(a.get_certificates_der_v3() + a.get_certificates_der_v2()
                        + [a.get_certificate_der(x)
                           for x in a.get_signature_names()])
            pkeys = set(a.get_public_keys_der_v3() + a.get_public_keys_der_v2())
```

5.3 测试报告解读

在等待大约几分钟后，Mobsf 会把分析结果展示出来。



因为涉及到敏感信息，也出于职业素养，这里我把 APP Information 都给隐藏掉。

我们可以看到， MobSF 的测试报告内容，是非常完善的。

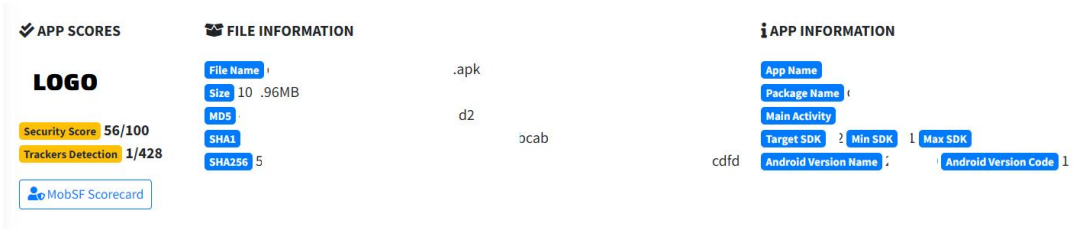
5.3.1 Information

Information：基本信息展示，APP Scores，File Information，APP Information 部分，

APP Scores ： 安全评分

File Information：文件名称，文件大小，文件加密类型；

APP Information： 应用名称，包名称，开发工具，安卓版本名称等；

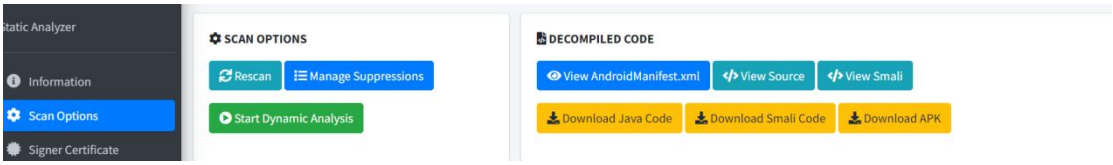


5.3.2 Scan Options

Scan Options： 扫描选项，有 SCAN OPTOINS 和 DECOMPLED CODE

SCAN OPTOINS：重新扫描，开始动态代码分析等；

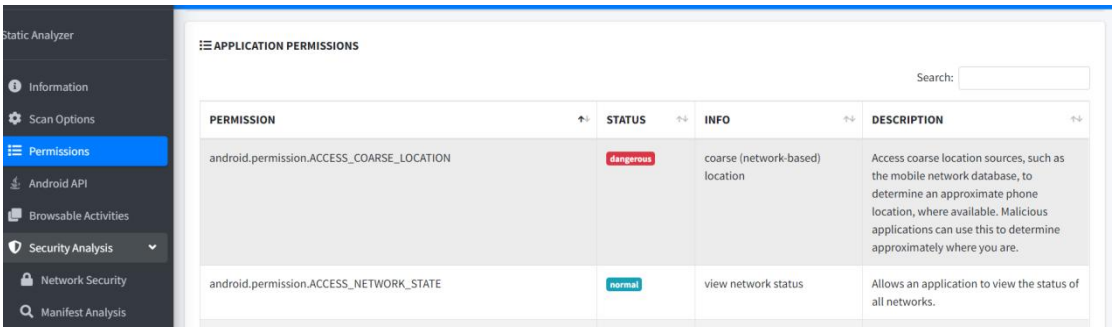
DECOMPLED CODE：查看 AndroidManifest.xml 文件、查询源码、查看 smali、下载源码。



5.3.3 Permissions

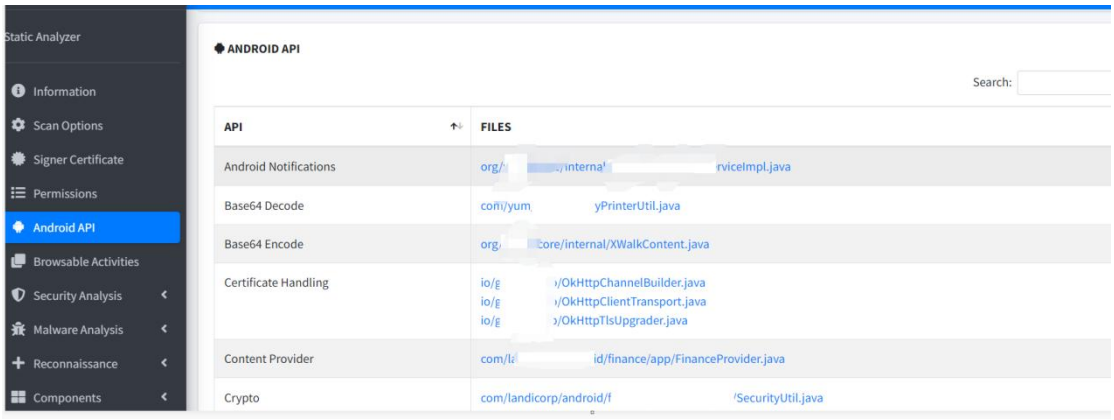
Permissions：许可认证。

这里会把所有的认证相关的漏洞，都列举出来。



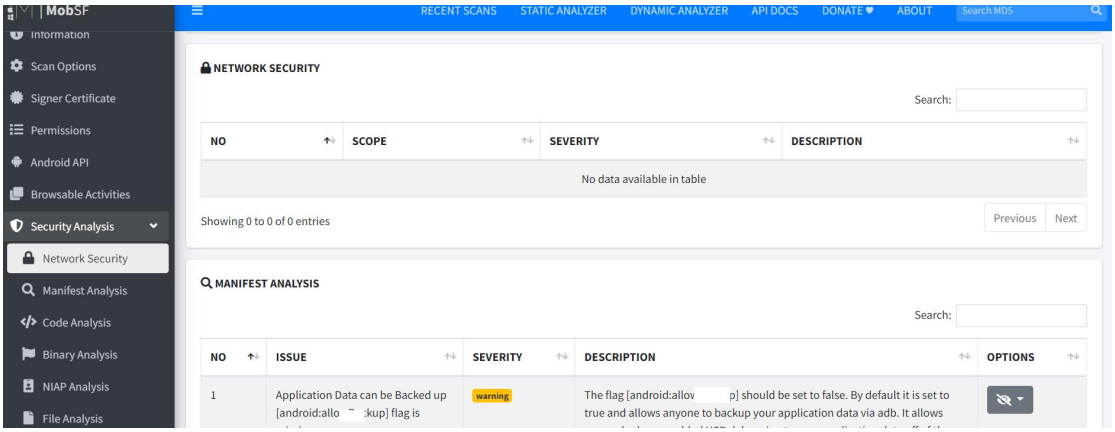
5.3.4 Android API

Android API：展示安卓的 API 信息：



5.3.5 Security Analysis

Security Analysis：安全分析，包含 Network Security，Manifest Analysis，Code Analysis，Binary Analysis，NIAP Analysis，File Analysis.



Network Security：网络安全

Manifest Analysis：清单分析，例如：是否允许备份(allowBackup)，正常是需要设为 false；

Code Analysis：代码分析，也就是整体代码静态扫描分析；

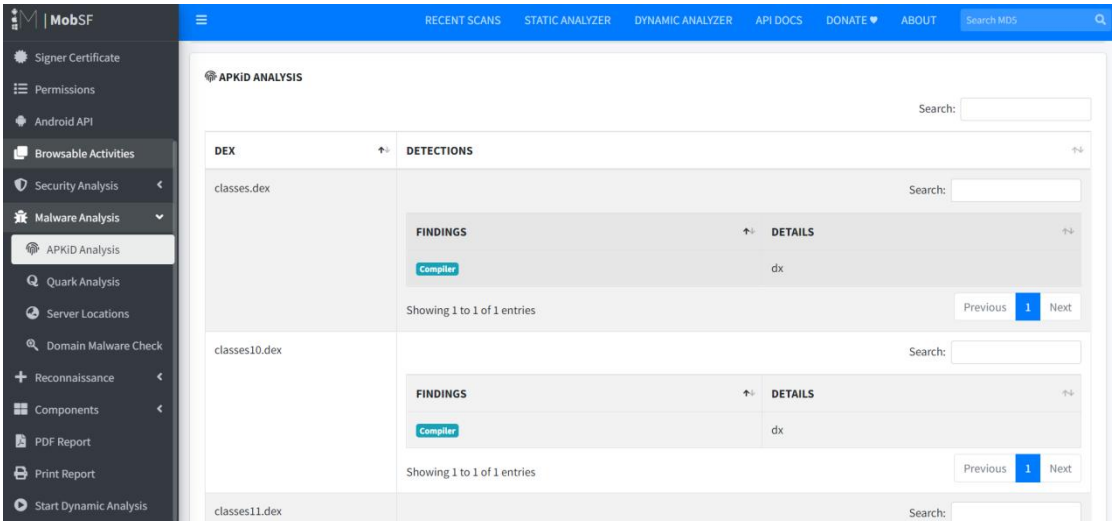
Binary Analysis：二进制分析

NIAP Analysis：NIAP 即(National Information Assurance Partnership)，在安全测试中都直接称为 NIAP 分析；

File Analysis：文件分析，这里的文件是指编码证书、秘钥文件等；

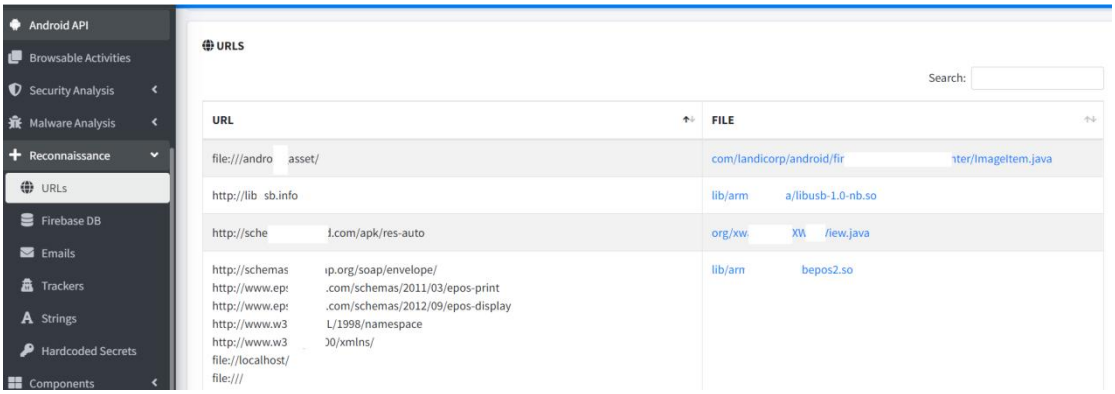
5.3.6 Malware Analysis

Malware Analysis：恶意软件分析，包含：APKiD Analysis，Quark Analysis，Server Locations，Domain Malware Check



5.3.7 Reconnaissance

Reconnaissance：侦查，探测，侦查的内容：URLs、Firebase DB、Emails、Trackers、Strings、Hardcode Secrets.



URLs：所有的 url 信息；

Firebase DB：数据库信息；

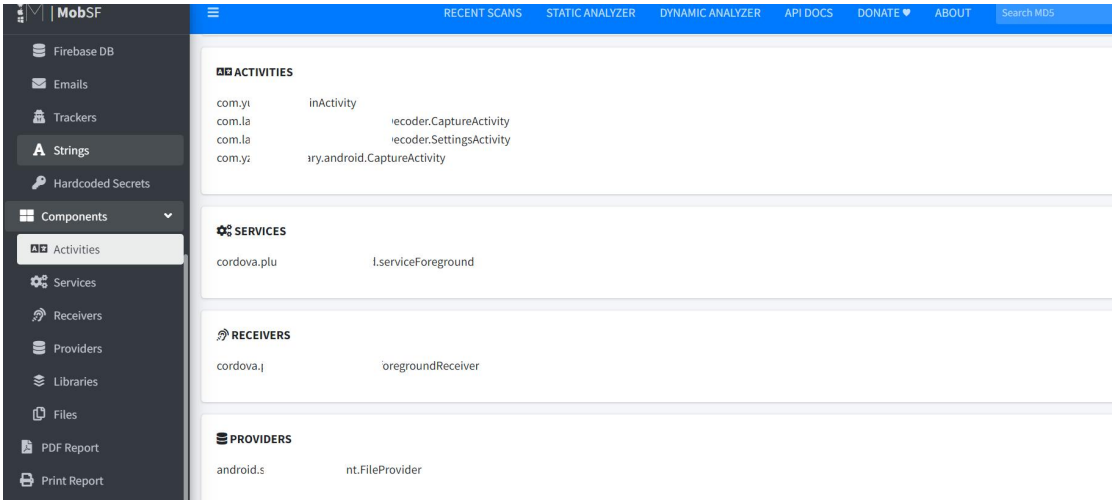
Emails：邮件信息；

Strings：字符串

Hardcode Secrets：硬编码机密；

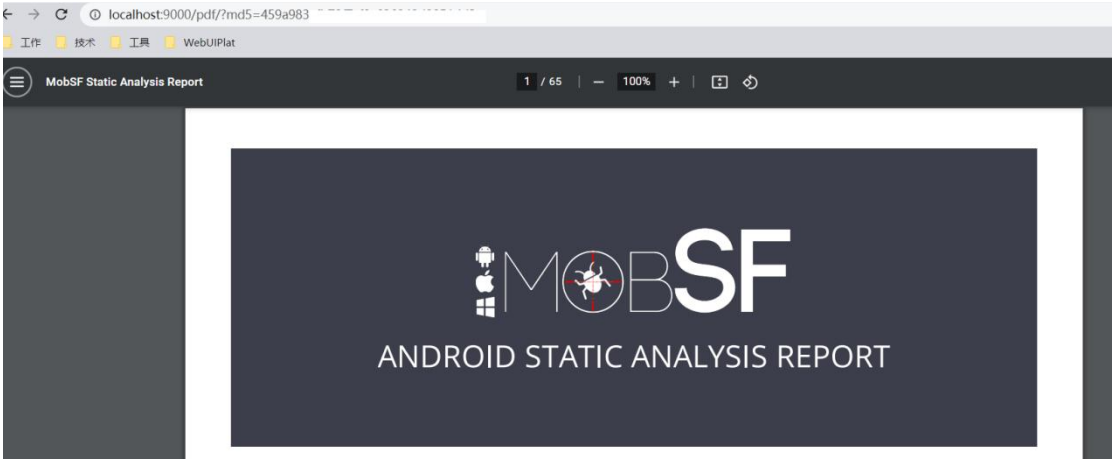
5.3.8 Components

Components：组件，主要包含 Activities、Services、Receivers、Providers、Libraries、Files.



5.3.9 PDF Report

测试报告以 PDF 的形式展示，如下图：



这也是必须 Wkhtmltopdf 的原因，否则，点击的时候，就会报错.如果要导出测试报告，直接点击 PDF Report 即可。

6、总结

看到这里，今天的分享差不多就该结束了。

回顾今天的内容，主要有：

-软件安全测试知识浅谈；

-软件安全测试工具了解；

-MobSF 服务的搭建、配置与操作

-MobSF 的原理分析与解读；

-MobSF 测试报告的内容解读；

今天的这篇内容，并没有涉及到 MobSF 的动态代码分析与扫描部分，主要考虑到，一次讲的内容太多，对学习效果不好，

所以，也就把静态代码分析与动态代码分析 分成两篇来讲解，这样，也有助于你对 MobSF 的学习与掌握。

今天也把代码逻辑贴出来，如果你看不懂，那可以把代码这部分给省略，

当然，针对我来说，还是希望你能看懂代码的，毕竟，这部分代码并不难，只要你有一点 python 基础的话。

如果确实一点 python 基础没有，那不要灰心，

天下无难事只怕有心人，

你要相信你自己能行，那一定可以能做到。

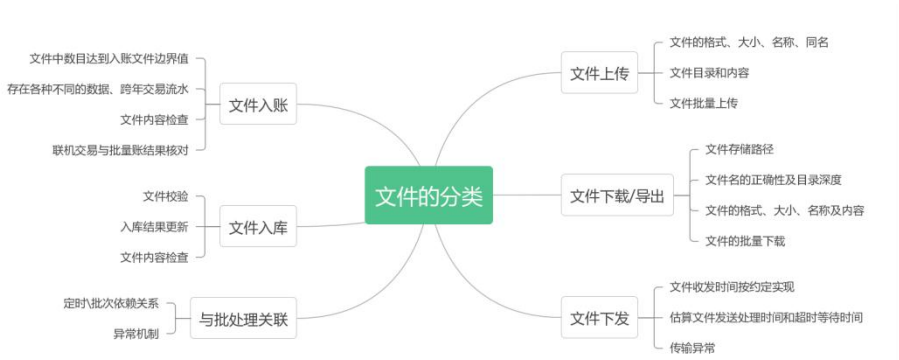
银行系统文件类的测试

◆作者：陆空

文件类的测试，它在测试中的关注度比较低，基本上没有形成固定的测试模式。在银行系统中，文件类测试却是非常重要的。从基本的文件上传、导出的测试，到跟批次相关的文件入账、下发等等都是重要的测试点，但却被很多测试人员忽略。导致需求或项目上线后，文件类程序出现的漏测率反而最高的，而漏测大部分的原因在于测试人员觉得简单，缺少考虑一些实际的测试场景。

一、文件测试的类型

在银行系统中，文件类测试相比其它的行业，它存在自己的特殊性，比如说，像文件入账、文件入库等，这些可能是其它的行业不存在的。



二、各种测试类型的测试点的分析

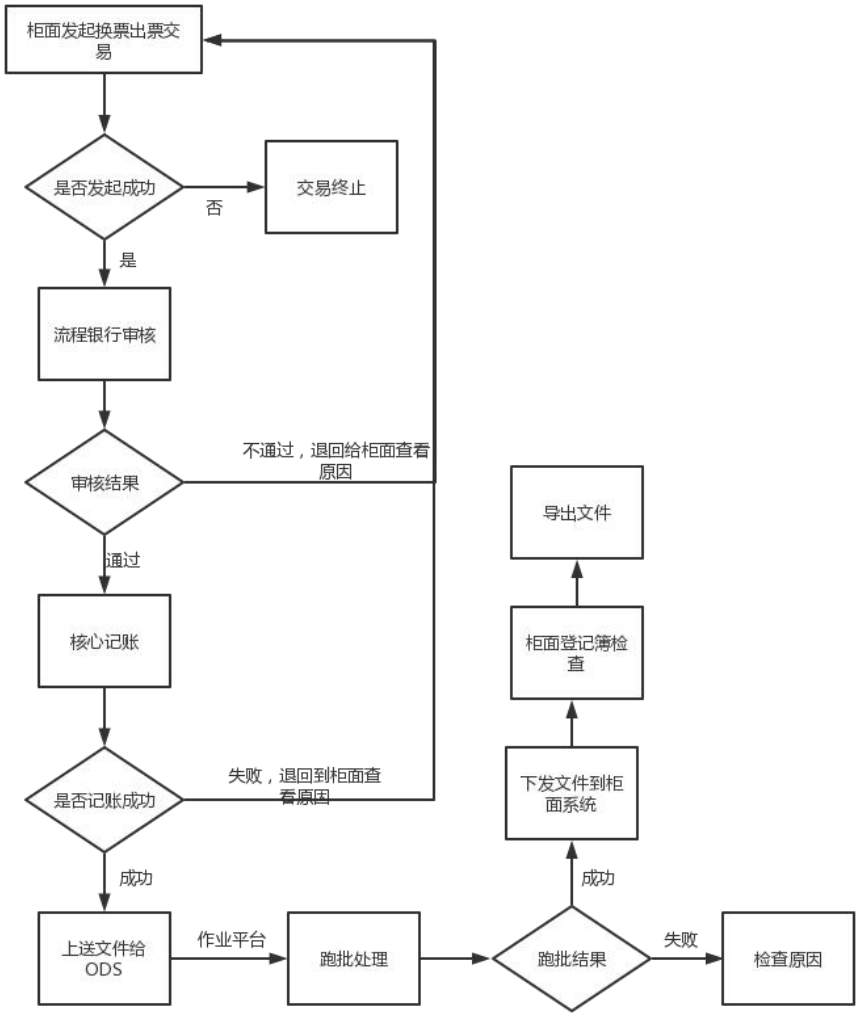
测试类型	关注点	关注的范围
文件上传	文件的格式	允许的格式是否均支持，只支持允许的文件格式。
	文件大小校验	文件大小/数据条数哈哈是否在文件提交前校验
	文件名校验	非法、合法字符校验（字母、中文、空格、特殊字符等）、长度边界检验。
	同名文件上传	是否支持同名文件长传，检查无重复数据处理。
	文件目录	检查上传后，文件路径是否正确
	文件内容	上传成功后，文件数据是否与实际一致
		上传的文件内容存在异常或是垃圾数据记录时，能正确处理
		上传的文件内容过程中断点或是其他异常操作导致上传中断，提示上传不成功
		上传的文件内容中断后重新上传
	短时间内多次上传文件名相同的文件	同 1 分钟内多次上次文件名相同的文件，检查处理是否正确
	批量上传	多个文件中，部分成功部分失败，能上传成功。 多个文件中，部分成功部分失败，文件数据正确 多个文件中，全部成功或全部失败，有正确提示。 多个文件中，全部成功或全部失败，文件数据正确。
	文件传输中断	可重新传输，并且不会产生重复数据或残缺数据导致系统或数据异常

文件下载/导出	文件存储路径	1、存储目录深度，深度超出提示 2、文件夹名称字符合法成功，非法提示
	文件名校验	各种字符类型是否支持，合法成功，非法提示
	文件内容及格式	1、下载/导出的格式是否正确 2、下载/导出的信息内容是否正确
	文件大小校验	文件大小/数据条数边界验证，符合边界要求，超出则提示或格式转换处理
	文件传输中断	可重新传输，并且不会产生重复数据或残缺数据导致系统或数据异常
文件入库	文件校验	检查批次入库前是否进行校验
	文件内容检查	1、文件中必填字段值为空 2、文件中非必填字段值为空（文件中附言与备注两项为空值
	入库中断	支持重新入库，并且重新入库后不会产生重复数据或残缺数据导致系统或数据异常
	文件入库更新	批次在处理大数据表时，脚本文件分批提交更新数据库
文件入账	入账文件存在各种数据	入账文件中有重复数据，入账失败，提示数据重复或自动过滤重复数 据，不重复入账。 入账文件中存在不同年，同月，同日入账数据，入账处理正确 入账文件中存在同年，不同月，那同日入账数据，入账处理正确。 入账文件中存在不同年，不同月，同日入账数据。 入账文件中存在跨年交易流水，检查程序是否会把入账文件中流水日期小于批次清算日期的交易剔除。 入账文件中存在空行，文件传输或转码正常处理
	入账文件数目	入账文件数目达到入账文件边界值，处理是否正确。 入账文件正常生成，但是入账记录条数为 0

	联机交易与批量入商户账结果核对	联机交易结束后，批量入商户账时检查入账总金额与联机交易发生总金额是否一致
	核心批次间发起记账文件	1、记账文件正常回盘，无错误挂账等账务问题。 2、核心批次作业正常，无出现断批的情况
	入账文件中部分相关金额超出系统设置阀值	1、入账文件中部分相关金额超出系统设置阀值 2、检查入账文件相关金额是否与系统设置的相关金额阀值有关， 3 并计算入账文件金额不能超出系统对应设置阀值的文件内容是否正确 4、核对客户交易金额是否存在虚增或虚减
文件下发	核心定时主动下发对账文件内容	核心定时下发的文件内容正常，不存在流水缺失的情况
	下发文件业务全流程检查	1、核心主动下发的文件，比如贷款及银承的各种登记簿 2、测试人员需要了解所测试文件的全业务流程，包括联机交易记档登记簿、核心取数生成文件、通过文件服务器下发文件至外围系统 3、检查各个过程中，记档各字段及文件下发时各字段是否正常，是否出现截取或乱码的情况

三、以银承业务中的换票出票的业务场景为例。

1、业务流程：柜面终端系统发起交易->流程银行审核->核心系统记账->通过 ODS 上送文件跑批->文件下发->银承登记簿



2、测试用例的设计

此场景的测试用例设计，是以上面的测试分析点为基准，再结合业务流程为出发点。再以业务流程为基础的用例，文件类的测试用例为重点。

业务流程类的测试用例设计：

用例编号	应用名称	模块	测试类型	测试检查要点	前置条件	测试步骤	预期结果
1	柜面终端系统	换票出票	功能测试	验证交易是否发起成功	客户需要存在银承汇票-纸票	1、登录柜面终端系统，录入客户的信息 2、点击提交按钮	1、交易成功 2、流程提交到流程银行平台中

2	流程银行平台系统	换票出票	功能测试	验证交易是否发起成功	客户需要存在银承汇票-纸票	1、登录流程银行，获取柜面发过来的交易 2、检查柜面上送的要素 3、点击审核通过按钮	1、获取到的交易与柜面发起的一致 2、平台的要素显示的与柜面录入的要素一致 3、审核通过，交易成功
3	核心系统	换票出票	功能测试	验证交易记账结果	交易发起成功	1、检查核心系统中此账号的信息 2、验证此交易的账务	1、此账号的票据记录成功 2、此账号的账务（工本费和手续费）记录正确
4	ODS 系统	核心系统文件	功能测试	验证抽取的文件	每日增量抽取	ODS 从核心系统抽取文件	ODS 抽取文件成功

文件类的测试用例设计：

用例编号	应用名称	模块	测试类型	测试检查要点	前置条件	测试步骤	预期结果
1	ODS 系统	入账文件	功能测试	验证文件的内容	交易流水	检查文件的内容与数量	1、文件的渠道显示正确 2、文件中的字段信息显示正确 3、文件中的数据量显示正确
2	集中作业平台	入账文件	功能测试	验证文件的格式及路径	交易流水	1、检查上送文件的格式 2、上送的文件的路径	1、文件的格式与预期结果一致 2、文件的名字中的日期显示与需求一致 3、文件的路径与需求一致

3	集中作业平台	批次处理	功能测试	验证文件入账的正确性	批次执行完成	1、文件中存在重复数据 2、文件中存在跨年交易流水 3、文件中存在空行	1、入账失败，提示数据重复或自动过滤重复数据，不重复入账。 2、入账成功，入账文件中流水日期小于批次清算日期的交易剔除。 2、入账成功，文件传输或转码正常处理。
4	集中作业平台	文件下发	功能测试	验证文件下发	批次执行完成	检查下发的文件	1、文件下发的路径符合需求 2、文件下发的名称显示正确 3、文件下发的内容显示正确
5	柜面终端系统	文件导出	功能测试	验证登记簿的结果	文件下发成功	1、进入银承登记簿 2、选择查询条件，点击查询按钮 3、点击导出按钮	1、银承登记簿查询出票据的结果正确 2、导出文件成功
6	柜面终端系统	文件导出	功能测试	验证导出文件内容及格式	文件下发成功	1、进入银承登记簿 2、点击导出按钮，导出文件 3、检查导出文件	1、文件的格式正确 2、文件的信息内容正确

四、收获

1、文件类的测试，有些测试点容易被测试人员忽略，比如说文件的命名、长度、路径，文件中的各种类型数据等。

2、银行的一条业务链包括着很多个业务系统，每个业务系统或多或少会有文件类的导入、导出或传输，且每个业务系统基本由不同的测试人员负责，每位测试人员对功能的关注点会有偏差，这就可能存在测试点遗漏，因此，对测试人员的业务场景的测试点培训就显示非常重要。

序号	依赖环境	安装	备注
1	Jdk Eclipse	安装包安装，注意 eclipse 和 jdk 版本号要一致，否则很容易启动 eclipse 时报错	安装完成后配置环境变量 新增 JAVA_HOME 为 jdk 安装路径 C:\Program Files\Java\jdk1.8.0_111 Path 中增加 %JAVA_HOME%\bin;%JAVA_HOME%\jre\bin CLASSPATH 增加值 为：;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar;
2	Androidsdk	解压即可	配置环境变量，同上 Path 中增加 sdk 的 tools 和 platform-tools 路径
3	Appiumdesktop	安装包安装	下载后直接安装就可以了
4	Appium-Python-Client	pip install Appium-Python-Client	进入命令行直接安装
5	夜神模拟器	安装包安装	下载后直接安装就可以了
6	Node.js	安装包安装	在 Windows 环境下，打开命令提示符，然后输入 node –v 可查看安装版本号

拓展学习

[1] 银行核心业务测试集锦

http://h.atstudy.com/yinhang/