

TPS 反编译适配系统适用于对目标机型进行反编译适配,将 tos 对原生 aosp 的改动,patch 到待适配目标机型中,并最终生成 ota 包。

一、运行环境

1. bash 环境,支持 mkdir、echo、find、md5sum、grep、basename、head、sed 等命令;
2. java 1.6 以上版本,并配置到 PATH 环境变量中。

二、目录结构

TPS	系统根目录
├─ Makefile	主 Makefile
├─ build	编译系统目录
│ └─ config	配置目录
│ └─ device_mk.template	设备 makefile 模板
│ └─ security	签名文件目录
│ └─ target_files_template	生成 OTA 包的 target files 模板
│ └─ tos_system_files.ignore	需要忽略的 tos 系统文件拷贝列表
│ └─ *.mk	makefile 文件
│ └─ tools	工具目录
└─ devices	设备目录
└─ tos	tos 机型(N5)目录
└─ i9500	目标机型目录（以 i9500 为例）
└─ Makefile	机型 makefile
└─ boot	boot.img 解开目录
└─ ota	机型的 OTA 解压目录
└─ ota.zip	机型的 OTA 包
└─ progress	流程控制目录
└─ smali	机型反编译后的原始 smali 及资源目录
└─ config	机型配置目录
└─ misc_info.txt	机型特定的 misc_info.txt
└─ recovery.fstab	机型特定的分区表
└─ remove_files.list	机型特定的精简文件列表
└─ patch	patch 工作目录
└─ config	patch 配置文件目录
└─ target	java 文件合并目标目录
└─ patch_failed.txt	合并失败的文件列表
└─ smali	smali patch 工作目录
└─ target	执行 method 合并的工作目录
└─ patch_failed.txt	合并失败的文件列表
└─ pack	make repack 工作目录
└─ package	打包 OTA 包目录
└─ i9500.zip	最终生成的 OTA 包



三、使用步骤（以 i9500 为例）

3.1 下载 TPS

```
$git clone https://github.com/TOSPlus/TPS
```

3.2 初始化环境变量

在 TPS 根目录下执行：

```
$source build/envsetup.sh
```

3.3 目标机型配置

1. 在 TPS 根目录下执行执行 **make create**，创建目标机型目录结构。

```
$make create brand=samsung device=i9500
```

其中 device 参数值可参考为 build.prop 中的 ro.product.model 字段值，不严格限制。

2. 将目标机型 **ota.zip** 包拷贝或链接至设备目录

```
$cp ota.zip TPS/device/i9500/
```

3. 在 TPS 根目录下执行执行 **make config**，对目标机型进行配置

```
$make config device=i9500
```

3.4 手动配置文件

1. 配置 TPS/device/i9500/Makefile

主要配置项主要包括：

DECOMPILE_PACKAGES	需要反编译的 jar/apk 文件，缺省包括 framework.jar、services.jar、android.policy.jar 、 framework2.jar、telephony-common.jar framework-res.apk
CUSTOM_RESOURCE_PACKAGE	原厂 ROM 自定义的资源 apk。默认搜索 ota/system/framework 下除 framework-res.apk 外的其它 apk 作为自定义的资源 apk
APKTOOL_JAR	缺省为 APKTOOL_JAR := \$(PORT_TOOLS)/apktool/apktool.jar
SUITABLE_DEVICES	指定生成 OTA 包时 updater-script 中指定 OTA 包所适用的机型列表
UNPACK_BOOTIMG_TOOL	解压 boot.img 的工具，默认为 built/tools/unpackbootimg.sh

若无特殊需求，均可采用默认配置。

2. 配置 misc_info.txt

在 i9500/config 目录下创建 misc_info.txt 文件，该文件负责描述目标机型分区大小等信息，用于最终 ota 打包。文件内容如下：

```
recovery_api_version=3
fstab_version=2
use_set_metadata=1
multistage_support=1
update_rename_support=1
fs_type=ext4
blocksize=131072
boot_size=8388608
recovery_size=8388608
system_size=2254438400
userdata_size=12738101248
cache_size=608174080
selinux_fc=BOOT/RAMDISK/file_contexts
```

其中红色参数部分需要根据目标机型的分区大小进行配置。

3. 配置 recovery.fstab

recovery.fstab 文件负责描述目标机型的分区信息，用于生成 OTA 包的 updater-script 脚本。可从目标机型的 recovery.img 中提取出来。提取方法如下：

```
$mkdir recovery
$~/TPS/build/tools/android/unpackbootimg -i recovery.img -o ./recovery
$cd recovery/
$cpio -i -d -m --no-absolute-filenames < recovery.img-ramdisk.gz
$gunzip -f recovery.img-ramdisk.gz
$cpio -i -d -m --no-absolute-filenames < recovery.img-ramdisk
$cd etc
$cp recovery.fstab ~/TPS/devices/SM-G900F/config/
```

4. 配置 remove_files.list

remove_files.list 文件用于配置原厂 ROM system 目录中需要精简的文件列表，格式如下：

```
app/PackageInstaller.apk
priv-app/SecContacts_OSup.apk
priv-app/SecNoteMyFiles.apk
app/SKMS_Agent_20130822_v0.952_UserKey.apk
app/SecurityProviderSEC.apk
app/SamsungAppsWidget.apk
app/AccessControl.apk
```

5. 配置 tos_system_files.ignore

tos_system_files.ignore，这个是要忽略拷贝的 tos 文件，目前来说，由于 media 注入有问题，需要配置以下两项：

```
bin/mediaserver_injector
lib/libmedia_jni.so
```

6. 配置 build.prop

将 config 目录下的 build.prop 模版的空白字段填上,可参考原厂 ota 包中 build.prop 进行配置, 如:

```
ro.qrom.beaconkey=0M000V5PH01B6QQD
ro.qrom.product.device=ja3g
ro.qrom.product.device.brand=samsung
ro.qrom.build.brand=tos
ro.qrom.build.os=android4.4.2
ro.qrom.otapath=/data/media/0
qrom_disposeIcon_enable=1
qrom_permission_enable=1
#
ro.qrom.build.version.snflag=ADRQRTOS
ro.qrom.build.version.snver=01
#need edit
ro.qrom.build.version.day=151028
ro.qrom.build.version.name=ADRQRTOS01_M
ro.qrom.build.version.number=01151028
ro.qrom.build.version.type=M
#need edit
ro.qrom.build.number=3
ro.qrom.build.lc=A1B2C3000D4E5F6
ro.qrom.build.lcid=99
#need edit
ro.qrom.build.date=Wed OCT 28 21:01:25 CST 2015
ro.qrom.build.date.utc=1446014851
ro.qrom.build.type=mod
```

另外, ro.qrom.build.version.number 是由 ro.qrom.build.version.snver 和 ro.qrom.build.version.day 拼起来的, 例如:

```
ro.qrom.build.version.snver=01
ro.qrom.build.version.day=151010
ro.qrom.build.version.number=01151010
```

7. 配置 boot.img

原厂的 boot.img 文件, 可以在 make prepare 的时候就放到 ota.zip 中, 也可配置到 override/BOOTABLE_IMAGES 目录下。若配置 override/BOOTABLE_IMAGES 目录下, 需要解 boot.img 得到 file_contexts 文件, 并拷贝到目标机型下的 config 目录中。解 boot.img 方法如下:

```
$mkdir boot_img
$~/TPS/build/tools/android/unpackbootimg -i boot.img -o boot_img/
$cd boot_img/
$gunzip -f boot.img-ramdisk.gz
$cpio -i -d --no-absolute-filenames <boot.img-ramdisk
```

```
Scp file_contexts ~/TPS/device/i9500/config/
```

3.5 执行 make prepare

解压 ota.zip，对 jar 和 apk 文件做 deodex 处理，将其反编译为 smali 文件。

3.6 执行 make patch

执行对 tos 和原厂 smali 文件的 patch。假如出现部分文件合并失败，会记录在 devices/i9500/patch/smali/target/patch_failed.txt 文件中，需要手动解决冲突文件。解决冲突方法有如下两种：

1) 源文件修改：

比如小米某一个方法多了一个参数，直接在 tos 的源文件（tos/smali）中把这个参数加上，重新 patch，搞定。再比如小米去除了某一个方法，在 mi4/smali 中添加这个方法的声明，重新 patch，搞定

2) 最终文件修改：

在 patch/smali/target 文件夹下直接修改 patch 失败的文件，改完之后 make repack，便会自动将所有 patch 的文件拷贝到 patch/smali/pack 文件夹下。

优先考虑第二种方式。

3.7 执行 make repack

检测 patch 是否成功，并准备好最终用于打包的 smali 文件。

3.8 执行 make package

准备 devices/i9500/package/target_files 目录，编译 jar 和 apk，重新打包 boot，最终在 package 目录下生成 ota 包。

四、部分常见问题解决方案

4.1 出现 “Permission denied……” 等相关错误

主要是脚本文件在下载过程中，文件的权限信息被修改，修改一下可执行权限便可。

4.2 出现 jar 文件 invalid 等错误

可能因为 jar 包为下载完全，或者部分软连接丢失。如将 baksmali.jar 软连接到 baksmali-2.0.3.jar 即可解决问题

4.3 每一个流程都会有对应的 clean

比如 make prepare 对应有 make clean-prepare，诸如此类有 make clean-patch、make clean-repack、make clean-package。若某一流程需要重新执行，则只需要先执行对应流程的 clean 便可。

4.4 生成的 update-script 中发现没有 symlink 的问题

最主要的原因是原厂的 ota 包不带符号链接导致不能生成 linkinfo.txt 文件，也就无法在刷机脚本中生成 symlink。解决方法就是在目标机型目录下，在 ota/META-INF/com/google/android/目录放置原厂的 updater-script 文件。TPS 便会解析原厂 updater-script 用于生成新包刷机脚本的 symlink。

4.5 如何使用 make syncpatch 对 framework 的更新进行处理

由于最新的 apk 更新都是跟 framework 的更新是同步的 假如只更新 apk, framework 不更新, 肯定会出问题。提供一个跟 framework 同步更新的简易方法, 增加了 make syncpatch 功能:

1.在目标机型目录下执行, make syncpatch, 会将 patch/target/smali/pack 目录拷贝到 temp/目录下临时备份

2.make clean-package

3.make package, 即可完成打包

4.6 make clean-package 略显蜗速, 怎么解?

初期设计主要考虑到不能删除开发者修改的文件, 所以会做大量检查, 占用时间。这个以后会做优化

你现在想快的话 可以直接手动删 rm -rf package 目录下的相关文件, 再执行 make clean-package

4.7 只改动了一个用于打包的 smali 文件, 放置在哪个目录, 如何编译新包??

patch/smali/pack 目录, 是用于打包的。放置在这个目录就可以了,

随后执行 make clean-package

make package 就可以完成打包

make package 会重新编译一次 jar 包, 不需要执行 make repack

4.8 TPS 自带的 apktool 工具无法直接使用, 本地有没有安装 apktool, 怎么解?

#反编译

```
java -jar ~/TPS/build/tools/apktool/apktool-2.0.0.jar d -f -t TPS_i9500 -o smali/Facelock.apk  
ota/system/app/FaceLock.apk
```

#编译 jar 和 apk

```
java -jar ~/TPS/build/tools/apktool/apktool-2.0.0.jar b android.policy.jar/
```

```
java -jar ~/TPS/build/tools/apktool/apktool-2.0.0.jar b Facelock.apk
```

4.9 patch/repack/package 的工作目录究竟是怎样的??

make patch patch 完毕的文件都将在 patch/smali/target 目录下

make repack, 便会自动将 patch 目录下的文件拷贝到 patch/smali/pack 文件夹下

make package 将 repack 目录的文件拷贝到 package 目录并进行编译打包