# Design and Development Manual for Hardening of Android Operating System

Document ID : Design and Development 1.4

Submitted By

C-DAC Hyderabad
Plot No. 6 & 7,
Hardware Park,
Sy No. 1/1,
Srisailam Highway,
Pahadi Shareef Via (Keshavagiri Post),
Hyderabad - 501510
Telangana(India)

Submitted To

ANURAG
DRDO, Ministry of Defence
Government of India

On

March 12, 2020

# Revision History

| Revision | Date | Author(s) | Description |
|---|---|---|---|
| 1.0 | 20.11.2018 | D M Vamsi | Created |
| 1.1 | 26.11.2018 | D M Vamsi | Added procedure for Unlocking bootloader,Flashing,FAQ's,Generation of Secret keys |
| 1.2 | 19.12.2018 | D M Vamsi | Updated Chapter 1 and 2 with respect to taimen build,Added Removal of Stock applications,Blocking of Third party application installations,Creation of New Application Domain,Integration of Whitelisted user applications,Resource Control,SELinux enforcement,Developer options protection |
| 1.2 | 13.02.2019 | D M Vamsi | Added procedure for Denying user space processes to write to /dev/mem or /dev/kmem ,Integrity Measurement Architecture, Observations during IMA implementation,Ulimit Configuration,Device Encryption,Fixed Path Execution of Executables, Scripts, Libraries and Applications, |

# Contents

# Chapter 1

# Downloading and Building

## 1.1 Objective

Objective of this chapter is to Download Android Framework source code corresponding to Google Pixel 2 XL Devices,building the same and flashing the generated custom images on Google Pixel 2 XL Devices.

## 1.2 Requirements

Before you download and build the Android source, ensure your system meets the following requirements

### 1.2.1 Hardware Requirements

Your development workstation should meet or exceed these hardware requirements:

- A 64-bit environment is required for Android 8.1.0 (Oreo)
- At least 100GB of free disk space to checkout the code and an extra 150GB to build it.If you conduct multiple builds or employ ccache, you will need even more space.
- If you are running Linux in a virtual machine, you need at least 16GB of RAM/swap.

### 1.2.2 Software Requirements

The Android Open Source Project (AOSP) master branch is traditionally developed and tested on Ubuntu Long Term Support (LTS) releases, but other distributions may be used.
If you are developing against the AOSP master branch, use operating systems: Ubuntu 14.04 (Trusty)
Key packages :

- Python 2.6 to 2.7 from python.org

- GNU Make 3.81 to 3.82 from gnu.org
- Git 1.7 or newer from git-scm.com

## 1.3 Establishing a Build Environment

This section describes how to set up your local work environment to build the Android source files. You must use Linux

### 1.3.1 Choosing branch

Some requirements for the build environment are determined by the version of the source code you plan to compile. For a full list of available branches, see `https://source.android.com/setup/start/build-numbers`

In our current project we are choosing :

- Build : OPM4.171019.016.B1
- Branch : android-8.1.0_r28
- Version : Oreo
- Supported devices : Pixel XL, Pixel, Pixel 2 XL, Pixel 2

### 1.3.2 Setting up a Linux build environment

The Android build is routinely tested in house on recent versions of Ubuntu LTS (14.04)

#### Installing the JDK

There are no available supported OpenJDK 8 packages for Ubuntu 14.04 directly Download the .deb packages for 64-bit architecture from old-releases.ubuntu.com:

- openjdk-8-jre-headless_8u45-b14-1_amd64.deb this file can be downloaded from `http://old-releases.ubuntu.com/ubuntu/pool/universe/o/openjdk-8/openjdk-8-jre-headless_8u45-b14-1_amd64.deb`
- openjdk-8-jre_8u45-b14-1_amd64.deb this file can be downloaded from `http://old-releases.ubuntu.com/ubuntu/pool/universe/o/openjdk-8/openjdk-8-jre_8u45-b14-1_amd64.deb`
- openjdk-8-jdk_8u45-b14-1_amd64.deb this file can be downloaded from `http://old-releases.ubuntu.com/ubuntu/pool/universe/o/openjdk-8/openjdk-8-jdk_8u45-b14-1_amd64.deb`

Install the packages by executing below commands:

```
$ sudo apt-get update
```

Run dpkg for each of the .deb files you downloaded. It may produce errors due to missing dependencies:

```
$ sudo dpkg -i {downloaded.deb file}
```

To fix missing dependencies:

```
$ sudo apt-get -f install
```

**Installing required packages (Ubuntu 14.04)**

The required packages can be installed using below command

```
$ sudo apt-get install git-core gnupg flex bison gperf build-essential zip curl
zlib1g-dev gcc-multilib g++-multilib libc6-dev-i386 lib32ncurses5-dev x11proto-core-dev
libx11-dev lib32z-dev libgl1-mesa-dev libxml2-utils xsltproc unzip
```

### 1.3.3 Downloading the Source

The Android source tree is located in a Git repository hosted by Google. The Git repository includes metadata for the Android source, including those related to changes to the source and the date they were made.

To Download the source code we required repo. The repo can be installed by executing below command :

```
$ sudo apt-get install phablet-tools
```

Run repo init to bring down the latest version of Repo with all its most recent bug fixes. You must specify a URL for the manifest, which specifies where the various repositories included in the Android source will be placed within your working directory.

```
$  repo init -u https://android.googlesource.com/platform/manifest -b android-8.1.0_r28
```

To pull down the Android source tree to your working directory from the repositories as specified in the default manifest, run

```
$ repo sync
```

The Android source files will be located in your working directory under their project names. The initial sync operation will take an hour or more to complete

### 1.3.4 Preparing to Build

The following instructions to build the Android source tree apply to all branches, including master. The basic sequence of build commands is as follows.

**Obtain proprietary binaries**

AOSP cannot be used from pure source code only and requires additional hardware-related proprietary libraries to run, such as for hardware graphics acceleration,etc. The below mentioned steps need to be followed to achieve all requirements The factory image zip of google Pixel 2 XL zip file need to be downloaded from below link

- `https://dl.google.com/dl/android/aosp/taimen-opm4.171019.016.b1-factory-f28b4a9b.zip`

The linuxbrew can be installable by executing below command

```
$ sh -c "$(curl -fsSL https://raw.githubusercontent.com/Linuxbrew/install/master/install.
sh)"
```

After successfull execution of above commands add below lines at the end of the `~/.bashrc` file and save it

```
export PATH=/home/linuxbrew/.linuxbrew/bin:$PATH
```

Then execute below command

```
$ source ~/.bashrc
```

Execute below commands to install ext4fuse

```
$ brew install libfuse
$ brew install ext4fuse
```

After successfull execution of above commands add below lines at the end of the `~/.bashrc` file and save it

```
export PATH=/home/linuxbrew/.linuxbrew/Cellar/ext4fuse/0.1.3/bin:$PATH
```

Then execute below command

```
$ source ~/.bashrc
```

Execute below command step by step to extract vendor blobs.

```
$ git clone https://github.com/anestisb/android-prepare-vendor.git
$ cd android-prepare-vendor/
$ mkdir pixel2xlvendor
$ sudo ./execute-all.sh -d taimen -i ~/Downloads/taimen-opm4.171019.016.b1-factory-f28b4a
9b.zip -b OPM4.171019.021.R1 -o pixel2xlvendor/
$ cd pixel2xlvendor/taimen/opm4.171019.021.r1
$ sudo chown -R cdac vendor/
```

Before execution of below command if any vendor folder exist in AOSP_Root delete the folder

```
$ sudo cp -R vendor/ ~/AOSP_Root/
```

### Cleanup

To ensure the newly installed binaries are properly taken into account after being extracted, delete the existing output of any previous build using:

```
$ cd AOSP_Root
$ make clobber
```

### 1.3.5 Build

#### Set up environment

Initialize the environment with the envsetup.sh script. Note that replacing source with . (a single dot) saves a few characters, and the short form is more commonly used in documentation.

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

or

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

#### Choose a target

Choose which target to build with lunch. The exact configuration can be passed as an argument.Based on our project requirement it should be

```
$ lunch aosp_taimen-user
```

This section is merely a summary to ensure setup is complete.

Build everything with make. GNU make can handle parallel tasks with a -jN argument, and it's common to use a number of tasks N that's between 1 and 2 times the number of hardware threads on the computer being used for the build. For example, on a dual-E5520 machine (2 CPUs, 4 cores per CPU, 2 threads per core), the fastest builds are made with commands between make -j16 and make -j32. In our project we can build the code by executing below command

```
$ make updatepackage -jN
```

The above command generates a file with name aosp_taimen-img-eng.root.zip in the following directory out/target/product/taimen

### 1.3.6 Unlocking the bootloader and To flash a system image

By Default the bootloader is locked and in order to flash a custom image we need to unlcok the bootloader. You can unlock the bootloader, but doing so deletes user data for privacy reasons. After unlocking, all data on the device is erased.

All Pixel devices released since 2014 have factory reset protection and require a multi-step process to unlock the bootloader.

To enable OEM unlocking on the Google Pixel 2 XL device:

- In Settings, tap About phone, then tap Build number seven (7) times.
- When you see the message "You are a developer", tap the back button.
- Tap Developer options and enable OEM unlocking and USB debugging. (If OEM unlocking is disabled, connect to the Internet so the device can check in at least once.)

Reboot into the bootloader and use fastboot to unlock it.

- Connect your device to your computer over USB.
- Start the device in fastboot mode with one of the following methods:

  [•] Using the adb tool: With the device powered on, execute

  ```
  $ adb reboot bootloader
  ```

  [•] Using a key combo: Turn the device off, then turn it on and immediately Press and hold Volume Down, then press and hold Power.

- unlock the device's bootloader by executing below command

  ```
  $ fastboot flashing unlock
  $ fastboot flashing unlock_critical
  ```

- Download all compatible requirements from below link to flash our customized operating system `https://dl.google.com/dl/android/aosp/taimen-opm4.171019.016.b1-factory-f28b4a9b.zip`

  We had already downloaded same form above url in previous module `1.3.4.PreparingtoBuild` under `Obtainproprietarybinaries` . We can utilize same .zip file here too, Instead of downloading again.

  ```
  $ unzip ~/Downloads/taimen-opm4.171019.016.b1-factory-f28b4a9b.zip
  $ cd taimen-opm4.171019.016.b1/
  $ ./flash-base.sh
  ```

- Finally flashing updated zip file from AOSP_Root directory.

  ```
  $ cd AOSP_Root
  $ fastboot -w update out/target/product/taimen/aosp_taimen-img-eng.root.zip
  ```

## 1.4 Frequently asked Questions

**My Android source code build end up with following error "GC overhead limit exceeded Try increasing heap size with java option '-Xmx<size>' "**

Once you're in the main source tree of AOSP run the following commands

```
$ export JACK_SERVER_VM_ARGUMENTS="-Dfile.encoding=UTF-8 -XX:+TieredCompilation -Xmx4g"
$ ./prebuilts/sdk/tools/jack-admin kill-server
$ ./prebuilts/sdk/tools/jack-admin start-server
```

**My Android source code build end up with following error "No Jack server running"**

Once you're in the main source tree of AOSP run the following commands

```
$ ./prebuilts/sdk/tools/jack-admin kill-server
$ ./prebuilts/sdk/tools/jack-admin start-server
```

# Chapter 2

# Android kernel compilation and Integrating to AOSP

## 2.1 Building Kernel

This chapter details how to build kernel corresponding to Google Pixel 2 XL device and incorporate into AOSP

### 2.1.1 Selecting a kernel

There exists different kernels for different devices, In our project context we require kernel compatible with taimen(Pixel 2 XL) device. Below information provides the name and locations of the kernel sources and binaries:

- Device : taimen
- Binary location : device/google/wahoo-kernel
- Source location : kernel/msm
- Build configuration : wahoo_defconfig

Below commands are useful for the cloning

```
$ cd AOSP_Root
$ . build/envsetup.sh
$ lunch aosp_taimen-user
$ cd
$ git clone https://android.googlesource.com/kernel/msm
$ cd msm
```

### 2.1.2 Prerequisities to Build the kernel

Building of the taimen kernel can be achieved by installing following prerequisities

```
$ sudo apt-get update
$ sudo apt-get install libssl-dev
$ sudo apt-get install device-tree-compiler
$ sudo apt install liblz4-tool
```

After successfull execution of above commands add below lines at the end of the `.bashrc` file

```
export PATH=AOSP_Root/prebuilts/gcc/linux-x86/aarch64/aarch64-linux-android-4.9/bin:$PA
TH
export ARCH=arm64
export CROSS_COMPILE=aarch64-linux-android-
```

Checkout the particular branch source code which you want to build using below command. In our project context we used the branch android-msm-wahoo-4.4-oreo-m4

```
$ git checkout -b android-msm-wahoo-4.4-oreo-m4 origin/android-msm-wahoo-4.4-oreo-m4
```

### 2.1.3 Enabling Touch modules in kernel

Identify `wahoo_defconfig` in the follwing path `Msm_Root/arch/arm64/configs` . Identify the following lines before modification

```
CONFIG_LGE_TOUCH_CORE=m
CONFIG_LGE_TOUCH_LGSIC_SW49408=m
CONFIG_TOUCHSCREEN_FTM4=m
```

Replace above identified lines with below lines

```
CONFIG_LGE_TOUCH_CORE=y
CONFIG_LGE_TOUCH_LGSIC_SW49408=y
CONFIG_TOUCHSCREEN_FTM4=y
```

### 2.1.4 Building Kernel

```
$ make wahoo_defconfig
$ make
```

After successful kernel build process completion, the required output file will be generated in below mentioned path with the file name Image.lz4-dtb
msm/arch/arm64/boot
copy or replace our custom build kernel output file Image.lz4-dtb into AOSP prebuild kernel path (i.e AOSP_ROOT/device/google/wahoo-kernel)

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with customized kernel.

# Chapter 3

# Generation of Secret keys required for AOSP Build process

## 3.1 Objective

Replacing all default keys with custom generated keys

## 3.2 Requirements

Generation of keys relevant to AOSP Build process can be achieved using make_-key tool.

### 3.2.1 Location of make_key tool in AOSP

The make_key is located in AOSP in below mentioned path:
AOSP_Root/development/tools

## 3.3 Generation of Release keys

The Android tree includes default-keys under build/target/product/security. Building an Android OS image using make will sign all .apk files using these default-keys based on application signature. So in our project we are trying to change these default-keys which are known to every one with our own custom secret keys to enhance Security.Execute following commands to generate custom Secret keys

```
$ cd AOSP_Root
$ subject='/C=IN/ST=India/L=Hyderabad/O=CDAC/OU=CDAC/CN=CDAC/emailAddress=cdac@cdac.in'
$ mkdir ~/.android-certs
```

```
$ for x in testkey platform shared media verity ; do ./development/tools/make_key ~/.android-certs/$x "$subject"; done
```

or follow bellow command to generate keys individually

```
$ development/tools/make_key platform '/C=IN/ST=India/L=Hyderabad/O=CDAC/OU=CDAC/CN=CDAC/e
mailAddress=cdac@cdac.in'
```

Repeat above same command with respect to testkey, shared,media and verity.
This results in the generation of following keys media.pk8,media.x509.pem,shared.pk8,shared.x509.pem, platform.pk8, platform.x509.pem,testkey.pk8,testkey.x509.pem,verity.pk8 and verity.x509.pem,

Once after successfully generation of keys copy these keys from .android-certs folder to build/target/product/security

## 3.4 Generation of verity_keys relevant for kernel compilation

Generate the verity public key with the help of generate_verity_key tool . The default location of tool is /AOSP_Root/out/host/linux-x86/bin . If it is not available in corresponding location generate the tool by executing below command from AOSP_Root

```
$ make -j16 generate_verity_key
```

Once after successful generation of generate_verity_key tool,use tool to generate verity public key in the format required for AOSP and as well generate verity keys which are required for kernel build process.

```
$ cd AOSP_Root
$ out/host/linux-x86/bin/generate_verity_key -convert
build/target/product/security/verity.x509.pem build/target/product/security/verity_key
```

The above command results a `verity_key.pub` in `build/target/product/security/` delete existing `verity_key` and rename the `verity_key.pub` to `verity_key` copy the verity.x509.pem from `AOSP_Root/build/target/product/security/` to `Msm_Root/certs/` . Generate our own custom `esl_key.pem` by following above key generation process.

Perform the subsection 2.1.4 in Chapter 2 Android kernel compilation and Integrating to AOSP to achieve new android operating system with customized kernel.

# Chapter 4

# Removal of Stock Applications

## 4.1 Objective

Removal of all unwanted stock applications from Android Platform

## 4.2 Identifying the targete applications

In our requirement we are considering only `Settings` application and rest all stock unwanted applications we are removing

### 4.2.1 aosp_base_telephony.mk

Identfiy aosp_base_telephony.mk in the following path `AOSP_Root/build/target/product/`. Replace complete file with following data

```
#
# Copyright 2013 The Android Open-Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#
$(call inherit-product, $(SRC_TARGET_DIR)/product/full_base_telephony.mk)

PRODUCT_PACKAGES += \
#    messaging
```

### 4.2.2 core.mk

Identfiy core.mk in the following path `AOSP_Root/build/target/product/`.
Replace complete file with following data

```
#
# Copyright (C) 2007 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Base configuration for communication-oriented android devices
# (phones, tablets, etc.).  If you want a change to apply to ALMOST ALL
# devices (including non-phones and non-tablets), modify
# core_minimal.mk instead. If you care about wearables, you need to modify
# core_tiny.mk in addition to core_minimal.mk.

PRODUCT_PACKAGES += \
    BasicDreams \
    BlockedNumberProvider \
    BookmarkProvider \
    BuiltInPrintService \
    CalendarProvider \
    CaptivePortalLogin \
    CertInstaller \
    DownloadProviderUi \
    ExternalStorageProvider \
    FusedLocation \
    InputDevices \
    KeyChain \
    Keyguard \
    LatinIME \
    Launcher2 \
    ManagedProvisioning \
    MtpDocumentsProvider \
    PicoTts \
    PacProcessor \
    libpac \
    PrintSpooler \
    PrintRecommendationService \
    ProxyHandler \
    Settings \
    SharedStorageBackup \
    StorageManager \
    Telecom \
    TeleService \
    VpnDialogs \
    vr \
    MmsService

# The set of packages whose code can be loaded by the system server.
PRODUCT_SYSTEM_SERVER_APPS += \
    FusedLocation \
    InputDevices \
    KeyChain \
    Telecom \

# The set of packages we want to force 'speed' compilation on.
PRODUCT_DEXPREOPT_SPEED_APPS += \

$(call inherit-product, $(SRC_TARGET_DIR)/product/core_base.mk)
```

### 4.2.3 generic_no_telephony.mk

Identfiy generic_no_telephony in the following path `AOSP_Root/build/target/product/`. Replace complete file with following data

```
#
# Copyright (C) 2007 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# This is a generic phone product that isn't specialized for a specific device.
# It includes the base Android platform.

PRODUCT_PACKAGES := \
    Bluetooth \
    BluetoothMidiService \
    MusicFX \
    NfcNci \
    OneTimeInitializer \
    Provision \
    SystemUI \
    SysuiDarkThemeOverlay \
    EasterEgg \
    WallpaperCropper

PRODUCT_PACKAGES += \
    clatd \
    clatd.conf \
    pppd \
    screenrecord

PRODUCT_PACKAGES += \
    librs_jni \
    libvideoeditor_jni \
    libvideoeditor_core \
    libvideoeditor_osal \
    libvideoeditor_videofilters \
    libvideoeditorplayer \

PRODUCT_PACKAGES += \
    audio.primary.default \
    local_time.default \
    vibrator.default \
    power.default

PRODUCT_COPY_FILES := \
        frameworks/av/media/libeffects/data/audio_effects.conf:system/etc/audio_effects.co

PRODUCT_PROPERTY_OVERRIDES += \
    ro.carrier=unknown

$(call inherit-product-if-exists, frameworks/base/data/fonts/fonts.mk)
$(call inherit-product-if-exists, external/google-fonts/dancing-script/fonts.mk)
$(call inherit-product-if-exists, external/google-fonts/carrois-gothic-sc/fonts.mk)
$(call inherit-product-if-exists, external/google-fonts/coming-soon/fonts.mk)
$(call inherit-product-if-exists, external/google-fonts/cutive-mono/fonts.mk)
$(call inherit-product-if-exists, external/noto-fonts/fonts.mk)
$(call inherit-product-if-exists, external/roboto-fonts/fonts.mk)
$(call inherit-product-if-exists, external/hyphenation-patterns/patterns.mk)
$(call inherit-product-if-exists, frameworks/base/data/keyboards/keyboards.mk)
$(call inherit-product-if-exists, frameworks/webview/chromium/chromium.mk)
$(call inherit-product, $(SRC_TARGET_DIR)/product/core.mk)

# Overrides
PRODUCT_BRAND := generic
```

```
PRODUCT_DEVICE := generic
PRODUCT_NAME := generic_no_telephony
```

### 4.2.4 telephony.mk

Identfiy telephony in the following path `AOSP_Root/build/target/product/`.
Replace complete file with following data

```
#
# Copyright (C) 2007 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# This is the list of product-level settings that are specific
# to products that have telephony hardware.

PRODUCT_PACKAGES := \
    CarrierConfig \
    CarrierDefaultApp \
    CallLogBackup \
    CellBroadcastReceiver \
    EmergencyInfo \
    rild

PRODUCT_COPY_FILES := \
```

### 4.2.5 aosp_taimen.mk

Identfiy aosp_taimen.mk in the following path `AOSP_Root/device/google/taimen/`.
Identify the following lines before modification

```
PRODUCT_PACKAGES += \
    Dialer \
    Launcher3 \
    WallpaperPicker
```

Replace above identified lines with below lines

```
PRODUCT_PACKAGES += \
    Launcher3 \
    WallpaperPicker
```

### 4.2.6 Messaging

Identfiy the `Messaging` folder in `AOSP_Root/packages/apps/`. Delete the complete folder

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with stock application removed.

# Chapter 5

# Blocking of Third party application Installations

## 5.1 Objective

Installed third party apps acts as the entry points for triggering malicious activity.If an app is hosted in Google play store it doesn't mean that it is a genuine app. Though google is having app-checking process to host genuine apps but fraudsters are cunning. As a result Google constantly removing fraud apps from the Android store( i.e such as fake antivirus , browser and games). The untrusted entry of app into our handheld device had lead to the miss use of hardware resources and data leak. The above points had made clear that more granular level of focus is essential to ensure trusted application installation flow control

## 5.2 Placing Applications in Domain

Every application in Android eco system must run in a specific application domain.Mapping of any application to specific domain is based on signature of the application . Default Android eco system has four major keys which are used to sign application: media,platform,shared and testkey . They are located in `AOSP_Root/build/target/product/security`. Any application which is signed other than these keys will come under default domain category which is untrusted_app domain.

We will discuss about some other important files which are having active role in placing Application in specific Domains

### 5.2.1 keys.conf

The actual magic doing the mapping from signature=@PLATFORM in mac_permissions.xml is in two locations `AOSP_Root/system/sepolicy/private/keys.conf` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/keys.conf` . This configuration file allows you to map a pem encoded x509 to an arbitrary string. The convention is to start them with @ The sample keys.conf file at `AOSP_Root/system/sepolicy/private/keys.conf` before customization.

```
#
# Maps an arbitrary tag [TAGNAME] with the string contents found in
# TARGET_BUILD_VARIANT. Common convention is to start TAGNAME with an @ and
# name it after the base file name of the pem file.
#
# Each tag (section) then allows one to specify any string found in
# TARGET_BUILD_VARIANT. Typcially this is user, eng, and userdebug. Another
# option is to use ALL which will match ANY TARGET_BUILD_VARIANT string.
#

[@PLATFORM]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/platform.x509.pem

[@MEDIA]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/media.x509.pem

[@SHARED]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/shared.x509.pem

# Example of ALL TARGET_BUILD_VARIANTS
[@RELEASE]
ENG       : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USER      : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USERDEBUG : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
```

The sample keys.conf file at `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/keys.conf` before customization.

```
#
# Maps an arbitrary tag [TAGNAME] with the string contents found in
# TARGET_BUILD_VARIANT. Common convention is to start TAGNAME with an @ and
# name it after the base file name of the pem file.
#
# Each tag (section) then allows one to specify any string found in
# TARGET_BUILD_VARIANT. Typcially this is user, eng, and userdebug. Another
# option is to use ALL which will match ANY TARGET_BUILD_VARIANT string.
#

[@PLATFORM]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/platform.x509.pem

[@MEDIA]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/media.x509.pem

[@SHARED]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/shared.x509.pem

# Example of ALL TARGET_BUILD_VARIANTS
[@RELEASE]
ENG       : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USER      : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USERDEBUG : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
```

### 5.2.2 The mac_permissions.xml file

The `AOSP_Root/system/sepolicy/private/mac_permissions.xml` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/mac_permissions.xml` files has a very confusing name. Expanded, the name is MAC permissions. However, its major mainline functionality is to map a signing key to a seinfo string The sample `AOSP_Root/system/sepolicy/private/mac_permissions.xml` file before customization.

```
<?xml version="1.0" encoding="utf-8"?>
<policy>

<!--

    * A signature is a hex encoded X.509 certificate or a tag defined in
      keys.conf and is required for each signer tag. The signature can
      either appear as a set of attached cert child tags or as an attribute.
    * A signer tag must contain a seinfo tag XOR multiple package stanzas.
    * Each signer/package tag is allowed to contain one seinfo tag. This tag
      represents additional info that each app can use in setting a SELinux security
      context on the eventual process as well as the apps data directory.
    * seinfo assignments are made according to the following rules:
      - Stanzas with package name refinements will be checked first.
      - Stanzas w/o package name refinements will be checked second.
      - The "default" seinfo label is automatically applied.

    * valid stanzas can take one of the following forms:

     // single cert protecting seinfo
     <signer signature="@PLATFORM" >
       <seinfo value="platform" />
     </signer>

     // multiple certs protecting seinfo (all contained certs must match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
       <seinfo value="platform" />
     </signer>

     // single cert protecting explicitly named app
     <signer signature="@PLATFORM" >
       <package name="com.android.foo">
         <seinfo value="bar" />
       </package>
     </signer>

     // multiple certs protecting explicitly named app (all certs must match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
       <package name="com.android.foo">
         <seinfo value="bar" />
       </package>
     </signer>
-->

    <!-- Platform dev key in AOSP -->
    <signer signature="@PLATFORM" >
      <seinfo value="platform" />
    </signer>

    <!-- Media key in AOSP -->
    <signer signature="@MEDIA" >
      <seinfo value="media" />
    </signer>

</policy>
```

The sample `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/mac_permissions.xml` file before customization.

```
<?xml version="1.0" encoding="utf-8"?>
<policy>

<!--

    * A signature is a hex encoded X.509 certificate or a tag defined in
      keys.conf and is required for each signer tag. The signature can
      either appear as a set of attached cert child tags or as an attribute.
    * A signer tag must contain a seinfo tag XOR multiple package stanzas.
    * Each signer/package tag is allowed to contain one seinfo tag. This tag
      represents additional info that each app can use in setting a SELinux security
      context on the eventual process as well as the apps data directory.
    * seinfo assignments are made according to the following rules:
      - Stanzas with package name refinements will be checked first.
      - Stanzas w/o package name refinements will be checked second.
      - The "default" seinfo label is automatically applied.

    * valid stanzas can take one of the following forms:

     // single cert protecting seinfo
     <signer signature="@PLATFORM" >
       <seinfo value="platform" />
     </signer>

     // multiple certs protecting seinfo (all contained certs must match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
       <seinfo value="platform" />
     </signer>

     // single cert protecting explicitly named app
     <signer signature="@PLATFORM" >
       <package name="com.android.foo">
         <seinfo value="bar" />
       </package>
     </signer>

     // multiple certs protecting explicitly named app (all certs must match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
       <package name="com.android.foo">
         <seinfo value="bar" />
       </package>
     </signer>
-->

    <!-- Platform dev key in AOSP -->
    <signer signature="@PLATFORM" >
      <seinfo value="platform" />
    </signer>

    <!-- Media key in AOSP -->
    <signer signature="@MEDIA" >
```

```
      <seinfo value="media" />
    </signer>

</policy>
```

### 5.2.3 seapp_contexts

The mapping of domain to application based on signature are get declared with the help of `AOSP_Root/system/sepolicy/private/seapp_contexts` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/seapp_contexts` files The sample `AOSP_Root/system/sepolicy/private/seapp_contexts` file before customization.

```
# Input selectors:
#       isSystemServer (boolean)
#       isEphemeralApp (boolean)
#       isV2App (boolean)
#       isOwner (boolean)
#       user (string)
#       seinfo (string)
#       name (string)
#       path (string)
#       isPrivApp (boolean)
#       minTargetSdkVersion (unsigned integer)
# isSystemServer=true can only be used once.
# An unspecified isSystemServer defaults to false.
# isEphemeralApp=true will match apps marked by PackageManager as Ephemeral
# isV2App=true will match apps in the v2 app sandbox.
# isOwner=true will only match for the owner/primary user.
# isOwner=false will only match for secondary users.
# If unspecified, the entry can match either case.
# An unspecified string selector will match any value.
# A user string selector that ends in * will perform a prefix match.
# user=_app will match any regular app UID.
# user=_isolated will match any isolated service UID.
# isPrivApp=true will only match for applications preinstalled in
#       /system/priv-app.
# minTargetSdkVersion will match applications with a targetSdkVersion
#       greater than or equal to the specified value. If unspecified,
#       it has a default value of 0.
# All specified input selectors in an entry must match (i.e. logical AND).
# Matching is case-insensitive.
#
# Precedence rules (see external/selinux/libselinux/src/android/android.c seapp_context_cmp()):
#       (1) isSystemServer=true before isSystemServer=false.
#       (2) Specified isEphemeralApp= before unspecified isEphemeralApp= boolean.
#       (3) Specified isV2App= before unspecified isV2App= boolean.
#       (4) Specified isOwner= before unspecified isOwner= boolean.
#       (5) Specified user= string before unspecified user= string.
#       (6) Fixed user= string before user= prefix (i.e. ending in *).
#       (7) Longer user= prefix before shorter user= prefix.
#       (8) Specified seinfo= string before unspecified seinfo= string.
#           ':' character is reserved and may not be used.
#       (9) Specified name= string before unspecified name= string.
#       (10) Specified path= string before unspecified path= string.
#       (11) Specified isPrivApp= before unspecified isPrivApp= boolean.
#       (12) Higher value of minTargetSdkVersion= before lower value of minTargetSdkVersion=
#              integer. Note that minTargetSdkVersion= defaults to 0 if unspecified.
#
# Outputs:
#       domain (string)
#       type (string)
#       levelFrom (string; one of none, all, app, or user)
#       level (string)
# Only entries that specify domain= will be used for app process labeling.
# Only entries that specify type= will be used for app directory labeling.
# levelFrom=user is only supported for _app or _isolated UIDs.
# levelFrom=app or levelFrom=all is only supported for _app UIDs.
# level may be used to specify a fixed level for any UID.
#
#
# Neverallow Assertions
# Additional compile time assertion checks can be added as well. The assertion
# rules are lines beginning with the keyword neverallow. Full support for PCRE
# regular expressions exists on all input and output selectors. Neverallow
# rules are never output to the built seapp_contexts file. Like all keywords,
# neverallows are case-insensitive. A neverallow is asserted when all key value
# inputs are matched on a key value rule line.
#

# only the system server can be in system_server domain
neverallow isSystemServer=false domain=system_server
neverallow isSystemServer="" domain=system_server

# system domains should never be assigned outside of system uid
neverallow user=((?!system).)* domain=system_app
neverallow user=((?!system).)* type=system_app_data_file

# anything with a non-known uid with a specified name should have a specified seinfo
neverallow user=_app name=.* seinfo=""
neverallow user=_app name=.* seinfo=default

# neverallow shared relro to any other domain
# and neverallow any other uid into shared_relro
neverallow user=shared_relro domain=((?!shared_relro).)*
neverallow user=((?!shared_relro).)* domain=shared_relro

# neverallow non-isolated uids into isolated_app domain
# and vice versa
neverallow user=_isolated domain=((?!isolated_app).)*
neverallow user=((?!_isolated).)* domain=isolated_app

# uid shell should always be in shell domain, however non-shell
# uid's can be in shell domain
neverallow user=shell domain=((?!shell).)*

# Ephemeral Apps must run in the ephemeral_app domain
neverallow isEphemeralApp=true domain=((?!ephemeral_app).)*

isSystemServer=true domain=system_server
user=system seinfo=platform domain=system_app type=system_app_data_file
user=bluetooth seinfo=platform domain=bluetooth type=bluetooth_data_file
user=nfc seinfo=platform domain=nfc type=nfc_data_file
user=radio seinfo=platform domain=radio type=radio_data_file
user=shared_relro domain=shared_relro
user=shell seinfo=platform domain=shell type=shell_data_file
user=_isolated domain=isolated_app levelFrom=user
user=_app seinfo=media domain=mediaprovider name=android.process.media type=app_data_file levelFrom=user
user=_app seinfo=platform domain=platform_app type=app_data_file levelFrom=user
user=_app isV2App=true isEphemeralApp=true domain=ephemeral_app type=app_data_file levelFrom=user
user=_app isPrivApp=true domain=priv_app type=app_data_file levelFrom=user
user=_app minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app domain=untrusted_app_25 type=app_data_file levelFrom=user
```

The sample `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/seapp_contexts` file before customization.

```
# Input selectors:
#       isSystemServer (boolean)
#       isEphemeralApp (boolean)
#       isV2App (boolean)
#       isOwner (boolean)
#       user (string)
#       seinfo (string)
#       name (string)
#       path (string)
#       isPrivApp (boolean)
#       minTargetSdkVersion (unsigned integer)
# isSystemServer=true can only be used once.
# An unspecified isSystemServer defaults to false.
# isEphemeralApp=true will match apps marked by PackageManager as Ephemeral
# isV2App=true will match apps in the v2 app sandbox.
# isOwner=true will only match for the owner/primary user.
# isOwner=false will only match for secondary users.
# If unspecified, the entry can match either case.
# An unspecified string selector will match any value.
# A user string selector that ends in * will perform a prefix match.
# user=_app will match any regular app UID.
# user=_isolated will match any isolated service UID.
# isPrivApp=true will only match for applications preinstalled in
#       /system/priv-app.
# minTargetSdkVersion will match applications with a targetSdkVersion
#       greater than or equal to the specified value. If unspecified,
#       it has a default value of 0.
# All specified input selectors in an entry must match (i.e. logical AND).
# Matching is case-insensitive.
#
# Precedence rules (see external/selinux/libselinux/src/android/android.c seapp_context_cmp()):
#       (1) isSystemServer=true before isSystemServer=false.
#       (2) Specified isEphemeralApp= before unspecified isEphemeralApp= boolean.
#       (3) Specified isV2App= before unspecified isV2App= boolean.
#       (4) Specified isOwner= before unspecified isOwner= boolean.
#       (5) Specified user= string before unspecified user= string.
#       (6) Fixed user= string before user= prefix (i.e. ending in *).
#       (7) Longer user= prefix before shorter user= prefix.
#       (8) Specified seinfo= string before unspecified seinfo= string.
#           ':' character is reserved and may not be used.
#       (9) Specified name= string before unspecified name= string.
#       (10) Specified path= string before unspecified path= string.
#       (11) Specified isPrivApp= before unspecified isPrivApp= boolean.
#       (12) Higher value of minTargetSdkVersion= before lower value of minTargetSdkVersion=
#              integer. Note that minTargetSdkVersion= defaults to 0 if unspecified.
#
# Outputs:
#       domain (string)
#       type (string)
#       levelFrom (string; one of none, all, app, or user)
#       level (string)
# Only entries that specify domain= will be used for app process labeling.
# Only entries that specify type= will be used for app directory labeling.
# levelFrom=user is only supported for _app or _isolated UIDs.
# levelFrom=app or levelFrom=all is only supported for _app UIDs.
# level may be used to specify a fixed level for any UID.
#
#
# Neverallow Assertions
# Additional compile time assertion checks can be added as well. The assertion
# rules are lines beginning with the keyword neverallow. Full support for PCRE
# regular expressions exists on all input and output selectors. Neverallow
# rules are never output to the built seapp_contexts file. Like all keywords,
# neverallows are case-insensitive. A neverallow is asserted when all key value
# inputs are matched on a key value rule line.
#

# only the system server can be in system_server domain
neverallow isSystemServer=false domain=system_server
neverallow isSystemServer="" domain=system_server

# system domains should never be assigned outside of system uid
neverallow user=((?!system).)* domain=system_app
neverallow user=((?!system).)* type=system_app_data_file

# anything with a non-known uid with a specified name should have a specified seinfo
neverallow user=_app name=.* seinfo=""
neverallow user=_app name=.* seinfo=default

# neverallow shared relro to any other domain
# and neverallow any other uid into shared_relro
neverallow user=shared_relro domain=((?!shared_relro).)*
neverallow user=((?!shared_relro).)* domain=shared_relro

# neverallow non-isolated uids into isolated_app domain
# and vice versa
neverallow user=_isolated domain=((?!isolated_app).)*
neverallow user=((?!_isolated).)* domain=isolated_app

# uid shell should always be in shell domain, however non-shell
# uid’s can be in shell domain
neverallow user=shell domain=((?!shell).)*

# Ephemeral Apps must run in the ephemeral_app domain
neverallow isEphemeralApp=true domain=((?!ephemeral_app).)*

isSystemServer=true domain=system_server
user=system seinfo=platform domain=system_app type=system_app_data_file
user=bluetooth seinfo=platform domain=bluetooth type=bluetooth_data_file
user=nfc seinfo=platform domain=nfc type=nfc_data_file
user=radio seinfo=platform domain=radio type=radio_data_file
user=shared_relro domain=shared_relro
user=shell seinfo=platform domain=shell type=shell_data_file
user=_isolated domain=isolated_app levelFrom=user
user=_app seinfo=platform domain=platform_app type=app_data_file levelFrom=user
user=_app isV2App=true isEphemeralApp=true domain=ephemeral_app type=app_data_file levelFrom=user
user=_app isV2App=true domain=untrusted_v2_app type=app_data_file levelFrom=user
user=_app isPrivApp=true domain=priv_app type=app_data_file levelFrom=user
user=_app minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app domain=untrusted_app_25 type=app_data_file levelFrom=user
```

## 5.3 Customizations to Restrict Third party application execution and installations

### 5.3.1 The mac_permissions.xml file

In this `AOSP_Root/system/sepolicy/private/mac_permissions.xml` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/mac_permissions.xml` file by default we had only platform and media keys only get mapped to a seinfo string. For our customization requirement we are mapping testkey and shared keys to seinfo strings. The below new lines need to be get added in both files.

```
<!-- Testkey in AOSP -->
<signer signature="@RELEASE" >
  <seinfo value="testkey" />
</signer>

<!-- Shared key in AOSP -->
<signer signature="@SHARED" >
  <seinfo value="shared" />
</signer>
```

### 5.3.2 seapp_contexts

The mapping of domain to application based on signature are get declared with the help of `AOSP_Root/System/sepolicy/private/seapp_contexts` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/seapp_contexts` files . The following lines in `AOSP_Root/System/sepolicy/private/seapp_contexts` before customizations made any application which does not having valid seinfo(default) to fell into untrusted app domain

```
user=_app minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app domain=untrusted_app_25 type=app_data_file levelFrom=user
```

The above two lines will get replaced with below lines which will enforce any application which is signed with media,shared and testkey into fell into untrusted domain . It also restrict all application executions which does not having a valid seinfo apart from which mentioned in `AOSP_Root/system/sepolicy/private/mac_permissions.xml`

```
user=_app seinfo=media minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app seinfo=shared minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app seinfo=testkey minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app seinfo=media domain=untrusted_app_25 type=app_data_file levelFrom=user
user=_app seinfo=shared domain=untrusted_app_25 type=app_data_file levelFrom=user
user=_app seinfo=testkey domain=untrusted_app_25 type=app_data_file levelFrom=user
```

The following lines in `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/seapp_contexts` before customizations made any application which does not having valid seinfo(default) to fell into untrusted app domain

```
user=_app isV2App=true domain=untrusted_v2_app type=app_data_file levelFrom=user
user=_app minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app domain=untrusted_app_25 type=app_data_file levelFrom=user
```

The above three lines will get replaced with below lines which will enforce any application which is signed with media,shared and testkey into fell into untrusted domain . It also restrict all application executions which does not having a valid seinfo apart from which mentioned in `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/mac_permissions.xml`

```
user=_app seinfo=media isV2App=true domain=untrusted_v2_app type=app_data_file levelFrom=user
user=_app seinfo=shared isV2App=true domain=untrusted_v2_app type=app_data_file levelFrom=user
user=_app seinfo=testkey isV2App=true domain=untrusted_v2_app type=app_data_file levelFrom=user
user=_app seinfo=media minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app seinfo=shared minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app seinfo=testkey minTargetSdkVersion=26 domain=untrusted_app type=app_data_file levelFrom=user
user=_app seinfo=media domain=untrusted_app_25 type=app_data_file levelFrom=user
user=_app seinfo=shared domain=untrusted_app_25 type=app_data_file levelFrom=user
user=_app seinfo=testkey domain=untrusted_app_25 type=app_data_file levelFrom=user
```

The above made changes successfully restrict Third party application execution in our Secured Anurag Platform using SEAndroid. Now we have to customize framework to completely restrict the installation of third party applications. To customize framework following file need to be targeted. `AOSP_Root//frameworks/base/services/core/java/com/android/server/pm/PackageManagerService.java` Identify below lines in PackageManagerService.java

```
if (mFoundPolicyFile) {
    SELinuxMMAC.assignSeInfoValue(pkg);
}
```

Replace above lines with below lines

```
if (mFoundPolicyFile) {
    SELinuxMMAC.assignSeInfoValue(pkg);
    if (pkg.applicationInfo.seInfo.contains("default")) {
        throw new PackageManagerException(PackageManager.INSTALL_FAILED_INVALID_APK, "Installation Blocked");
    }
}
```

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with capability of blocking third party application installations.

Chapter 6

# Creation of New Application Domain

## 6.1 Objective

## 6.2 Creation of Application Domain

To have a domain based resource control we are creating three new application domains namely anuragtrusted_app,anuragtrustedtwo_app and anuragtrustedthree_-app. To create a new domain the following directories/files need to be targated in AOSP_Root

```
AOSP_Root/system/sepolicy/private
AOSP_Root/system/sepolicy/prebuilts/api/26.0/private
AOSP_Root/build/target/product/security
AOSP_Root/system/sepolicy/private/app.te
AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/app.te
AOSP_Root/system/sepolicy/private/seapp_contexts
AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/seapp_contexts
AOSP_Root/system/sepolicy/private/mac_permissions.xml
AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/mac_permissions.
xml
AOSP_Root/system/sepolicy/private/keys.conf
AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/keys.conf
```

### 6.2.1 Security

By default, there are four different keys in the current Android source tree: platform, shared, media, and testkey (releasekey for release builds). These keys plays a keyrole in making any application to fall in specific domain . In our requirement we require three different key pairs for three domains anuragtrusted_-app,anuragtrustedtwo_app and anuragtrustedthree_app. Generate three key pairs with name anurag.pk8,anurag.x509.pem,anuragtwo.pk8,anuragtwo.x509.pem,anuragthree.pk8 and anuragthree.x509.pem. Copy all these three key pairs into the location of `AOSP_Root/build/target/product/security`

### 6.2.2 keys.conf

These two `AOSP_Root/system/sepolicy/private/keys.conf` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/keys.conf` configuration files allows you to map a pem encoded x509 to an arbitrary string. As per our customization requirements we need to map for our anuragtrusted_app,anuragtrustedtwo_app and anuragtrustedthree_app

Below lines depicts keys.conf file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` before mapping anurag pem files

```
#
# Maps an arbitrary tag [TAGNAME] with the string contents found in
# TARGET_BUILD_VARIANT. Common convention is to start TAGNAME with an @ and
# name it after the base file name of the pem file.
#
# Each tag (section) then allows one to specify any string found in
# TARGET_BUILD_VARIANT. Typcially this is user, eng, and userdebug. Another
# option is to use ALL which will match ANY TARGET_BUILD_VARIANT string.
#

[@PLATFORM]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/platform.x509.pem

[@MEDIA]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/media.x509.pem

[@SHARED]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/shared.x509.pem

# Example of ALL TARGET_BUILD_VARIANTS
[@RELEASE]
ENG       : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USER      : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USERDEBUG : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
```

Below lines depicts keys.conf file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` after mapping anurag pem files

```
#
# Maps an arbitrary tag [TAGNAME] with the string contents found in
# TARGET_BUILD_VARIANT. Common convention is to start TAGNAME with an @ and
# name it after the base file name of the pem file.
#
# Each tag (section) then allows one to specify any string found in
# TARGET_BUILD_VARIANT. Typcially this is user, eng, and userdebug. Another
# option is to use ALL which will match ANY TARGET_BUILD_VARIANT string.
#

[@PLATFORM]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/platform.x509.pem

[@MEDIA]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/media.x509.pem

[@SHARED]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/shared.x509.pem

[@ANURAG]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/anurag.x509.pem

[@ANURAGTWO]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/anuragtwo.x509.pem

[@ANURAGTHREE]
ALL : $DEFAULT_SYSTEM_DEV_CERTIFICATE/anuragthree.x509.pem


# Example of ALL TARGET_BUILD_VARIANTS
[@RELEASE]
ENG       : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USER      : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
USERDEBUG : $DEFAULT_SYSTEM_DEV_CERTIFICATE/testkey.x509.pem
```

### 6.2.3 mac_permissions.xml

`AOSP_Root/system/sepolicy/private/mac_permissions.xml` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/mac_permissions.xml` assigns a seinfo tag to apps based on their signature and optionally their package name. The seinfo tag can then be used as a key in the seapp_contexts file to assign a specific label to all apps with that seinfo tag. The `mac_permission.xml` in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` before adding anurag new domain seinfo tags

```
<?xml version="1.0" encoding="utf-8"?>
<policy>

<!--

    * A signature is a hex encoded X.509 certificate or a tag defined in
      keys.conf and is required for each signer tag. The signature can
      either appear as a set of attached cert child tags or as an attribute.
    * A signer tag must contain a seinfo tag XOR multiple package stanzas.
    * Each signer/package tag is allowed to contain one seinfo tag. This tag
      represents additional info that each app can use in setting a SELinux security
      context on the eventual process as well as the apps data directory.
    * seinfo assignments are made according to the following rules:
      - Stanzas with package name refinements will be checked first.
      - Stanzas w/o package name refinements will be checked second.
      - The "default" seinfo label is automatically applied.

    * valid stanzas can take one of the following forms:

     // single cert protecting seinfo
     <signer signature="@PLATFORM" >
       <seinfo value="platform" />
     </signer>

     // multiple certs protecting seinfo (all contained certs must match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
       <seinfo value="platform" />
     </signer>

     // single cert protecting explicitly named app
     <signer signature="@PLATFORM" >
       <package name="com.android.foo">
         <seinfo value="bar" />
       </package>
     </signer>

     // multiple certs protecting explicitly named app (all certs must match)
     <signer>
       <cert signature="@PLATFORM1"/>
       <cert signature="@PLATFORM2"/>
       <package name="com.android.foo">
         <seinfo value="bar" />
       </package>
     </signer>
-->

    <!-- Platform dev key in AOSP -->
    <signer signature="@PLATFORM" >
      <seinfo value="platform" />
    </signer>

    <!-- Media key in AOSP -->
    <signer signature="@MEDIA" >
      <seinfo value="media" />
    </signer>

    <!-- Platform dev key in AOSP -->
    <signer signature="@RELEASE" >
      <seinfo value="testkey" />
    </signer>

    <!-- Media key in AOSP -->
    <signer signature="@SHARED" >
      <seinfo value="shared" />
    </signer>

</policy>
```

The `mac_permission.xml` in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` after adding anurag new domain seinfo tags

```
<?xml version="1.0" encoding="utf-8"?>
<policy>

<!--

    * A signature is a hex encoded X.509 certificate or a tag defined in
      keys.conf and is required for each signer tag. The signature can
      either appear as a set of attached cert child tags or as an attribute.
```

```
     * A signer tag must contain a seinfo tag XOR multiple package stanzas.
     * Each signer/package tag is allowed to contain one seinfo tag. This tag
       represents additional info that each app can use in setting a SELinux security
       context on the eventual process as well as the apps data directory.
     * seinfo assignments are made according to the following rules:
       - Stanzas with package name refinements will be checked first.
       - Stanzas w/o package name refinements will be checked second.
       - The "default" seinfo label is automatically applied.

     * valid stanzas can take one of the following forms:

      // single cert protecting seinfo
      <signer signature="@PLATFORM" >
        <seinfo value="platform" />
      </signer>

      // multiple certs protecting seinfo (all contained certs must match)
      <signer>
        <cert signature="@PLATFORM1"/>
        <cert signature="@PLATFORM2"/>
        <seinfo value="platform" />
      </signer>

      // single cert protecting explicitly named app
      <signer signature="@PLATFORM" >
        <package name="com.android.foo">
          <seinfo value="bar" />
        </package>
      </signer>

      // multiple certs protecting explicitly named app (all certs must match)
      <signer>
        <cert signature="@PLATFORM1"/>
        <cert signature="@PLATFORM2"/>
        <package name="com.android.foo">
          <seinfo value="bar" />
        </package>
      </signer>
-->

    <!-- Platform dev key in AOSP -->
    <signer signature="@PLATFORM" >
      <seinfo value="platform" />
    </signer>

    <!-- Media key in AOSP -->
    <signer signature="@MEDIA" >
      <seinfo value="media" />
    </signer>

    <!-- Platform dev key in AOSP -->
    <signer signature="@RELEASE" >
      <seinfo value="testkey" />
    </signer>

    <!-- Media key in AOSP -->
    <signer signature="@SHARED" >
      <seinfo value="shared" />
    </signer>

    <signer signature="@ANURAG" >
      <seinfo value="anurag" />
    </signer>

    <signer signature="@ANURAGTWO" >
      <seinfo value="anuragtwo" />
    </signer>

    <signer signature="@ANURAGTHREE" >
      <seinfo value="anuragthree" />
    </signer>

</policy>
```

### 6.2.4 seapp_contexts

The mapping of domain to application based on signature are get declared with the help of `AOSP_Root/System/sepolicy/private/seapp_contexts` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/seapp_contexts` . The Below mentioned new lines need to be added at end of `seapp_contexts` file in both directories `AOSP_Root/system/sepolicy/private` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private`

```
user=_app seinfo=anurag domain=anuragtrusted_app type=app_data_file levelFrom=user
user=_app seinfo=anuragtwo domain=anuragtrustedtwo_app type=app_data_file levelFrom=user
user=_app seinfo=anuragthree domain=anuragtrustedthree_app type=app_data_file levelFrom=user
```

### 6.2.5 anurgtrusted_app,anuragtrustedtwo_app and anuragtrustedthree_app

We are creting new sepolicy in both directories `AOSP_Root/System/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private` with the names `anuragtrusted_app` ,`anuragtrustedtwo_app` and `anuragtrustedthree_app`.

The Below following lines getting added to `AOSP_Root/System/sepolicy/private/anuragtrusted_app` by taking `AOSP_Root/system/sepolicy/private/platform_app` as refernece.

```
###
### Apps signed with the anuragtrusted_app key.
###
type anuragtrusted_app, domain;
typeattribute anuragtrusted_app coredomain;


app_domain(anuragtrusted_app)

# Access the network.
net_domain(anuragtrusted_app)
# Access bluetooth.
bluetooth_domain(anuragtrusted_app)
# Read from /data/local/tmp or /data/data/com.android.shell.
allow anuragtrusted_app shell_data_file:dir search;
allow anuragtrusted_app shell_data_file:file { open getattr read };
allow anuragtrusted_app icon_file:file { open getattr read };

# ASEC
allow anuragtrusted_app asec_apk_file:dir create_dir_perms;
allow anuragtrusted_app asec_apk_file:file create_file_perms;

# Access to /data/media.
allow anuragtrusted_app media_rw_data_file:dir create_dir_perms;
allow anuragtrusted_app media_rw_data_file:file create_file_perms;

# Write to /cache.
allow anuragtrusted_app cache_file:dir create_dir_perms;
allow anuragtrusted_app cache_file:file create_file_perms;

# Direct access to vold-mounted storage under /mnt/media_rw
# This is a performance optimization that allows anuragtrusted_app apps to bypass the FUSE layer
allow anuragtrusted_app mnt_media_rw_file:dir r_dir_perms;
allow anuragtrusted_app vfat:dir create_dir_perms;
allow anuragtrusted_app vfat:file create_file_perms;

# com.android.systemui
allow anuragtrusted_app rootfs:dir getattr;

allow anuragtrusted_app audioserver_service:service_manager find;
allow anuragtrusted_app cameraserver_service:service_manager find;
allow anuragtrusted_app drmserver_service:service_manager find;
allow anuragtrusted_app mediaserver_service:service_manager find;
allow anuragtrusted_app mediametrics_service:service_manager find;
allow anuragtrusted_app mediaextractor_service:service_manager find;
allow anuragtrusted_app mediacodec_service:service_manager find;
allow anuragtrusted_app mediadrmserver_service:service_manager find;
allow anuragtrusted_app persistent_data_block_service:service_manager find;
allow anuragtrusted_app radio_service:service_manager find;
allow anuragtrusted_app surfaceflinger_service:service_manager find;
allow anuragtrusted_app timezone_service:service_manager find;
allow anuragtrusted_app app_api_service:service_manager find;
allow anuragtrusted_app system_api_service:service_manager find;
allow anuragtrusted_app vr_manager_service:service_manager find;

# Access to /data/preloads
allow anuragtrusted_app preloads_data_file:file r_file_perms;
allow anuragtrusted_app preloads_data_file:dir r_dir_perms;
allow anuragtrusted_app preloads_media_file:file r_file_perms;
allow anuragtrusted_app preloads_media_file:dir r_dir_perms;

###
### Neverallow rules
###

# app domains which access /dev/fuse should not run as anuragtrusted_app
neverallow anuragtrusted_app fuse_device:chr_file *;
```

The Below following lines getting added to `AOSP_Root/System/sepolicy/private/anuragtrustedtwo_app` by taking `AOSP_Root/system/sepolicy/private/platform_app` as refernece.

```
###
```

```
### Apps signed with the anuragtrustedtwo_app key.
###
type anuragtrustedtwo_app, domain;
typeattribute anuragtrustedtwo_app coredomain;


app_domain(anuragtrustedtwo_app)

# Access the network.
net_domain(anuragtrustedtwo_app)
# Access bluetooth.
bluetooth_domain(anuragtrustedtwo_app)
# Read from /data/local/tmp or /data/data/com.android.shell.
allow anuragtrustedtwo_app shell_data_file:dir search;
allow anuragtrustedtwo_app shell_data_file:file { open getattr read };
allow anuragtrustedtwo_app icon_file:file { open getattr read };

# ASEC
allow anuragtrustedtwo_app asec_apk_file:dir create_dir_perms;
allow anuragtrustedtwo_app asec_apk_file:file create_file_perms;

# Access to /data/media.
allow anuragtrustedtwo_app media_rw_data_file:dir create_dir_perms;
allow anuragtrustedtwo_app media_rw_data_file:file create_file_perms;

# Write to /cache.
allow anuragtrustedtwo_app cache_file:dir create_dir_perms;
allow anuragtrustedtwo_app cache_file:file create_file_perms;

# Direct access to vold-mounted storage under /mnt/media_rw
# This is a performance optimization that allows anuragtrustedtwo_app apps to bypass the FUSE layer
allow anuragtrustedtwo_app mnt_media_rw_file:dir r_dir_perms;
allow anuragtrustedtwo_app vfat:dir create_dir_perms;
allow anuragtrustedtwo_app vfat:file create_file_perms;

# com.android.systemui
allow anuragtrustedtwo_app rootfs:dir getattr;

allow anuragtrustedtwo_app audioserver_service:service_manager find;
allow anuragtrustedtwo_app cameraserver_service:service_manager find;
allow anuragtrustedtwo_app drmserver_service:service_manager find;
allow anuragtrustedtwo_app mediaserver_service:service_manager find;
allow anuragtrustedtwo_app mediametrics_service:service_manager find;
allow anuragtrustedtwo_app mediaextractor_service:service_manager find;
allow anuragtrustedtwo_app mediacodec_service:service_manager find;
allow anuragtrustedtwo_app mediadrmserver_service:service_manager find;
allow anuragtrustedtwo_app persistent_data_block_service:service_manager find;
allow anuragtrustedtwo_app radio_service:service_manager find;
allow anuragtrustedtwo_app surfaceflinger_service:service_manager find;
allow anuragtrustedtwo_app timezone_service:service_manager find;
allow anuragtrustedtwo_app app_api_service:service_manager find;
allow anuragtrustedtwo_app system_api_service:service_manager find;
allow anuragtrustedtwo_app vr_manager_service:service_manager find;

# Access to /data/preloads
allow anuragtrustedtwo_app preloads_data_file:file r_file_perms;
allow anuragtrustedtwo_app preloads_data_file:dir r_dir_perms;
allow anuragtrustedtwo_app preloads_media_file:file r_file_perms;
allow anuragtrustedtwo_app preloads_media_file:dir r_dir_perms;


###
### Neverallow rules
###

# app domains which access /dev/fuse should not run as anuragtrustedtwo_app
neverallow anuragtrustedtwo_app fuse_device:chr_file *;
```

The Below following lines getting added to `AOSP_Root/System/sepolicy/private/anuragtrustedthree_app` by taking `AOSP_Root/system/sepolicy/private/platform_app` as refernece.

```
###
### Apps signed with the anuragtrustedthree_app key.
###
type anuragtrustedthree_app, domain;
typeattribute anuragtrustedthree_app coredomain;


app_domain(anuragtrustedthree_app)

# Access the network.
net_domain(anuragtrustedthree_app)
# Access bluetooth.
bluetooth_domain(anuragtrustedthree_app)
# Read from /data/local/tmp or /data/data/com.android.shell.
allow anuragtrustedthree_app shell_data_file:dir search;
allow anuragtrustedthree_app shell_data_file:file { open getattr read };
allow anuragtrustedthree_app icon_file:file { open getattr read };

# ASEC
allow anuragtrustedthree_app asec_apk_file:dir create_dir_perms;
allow anuragtrustedthree_app asec_apk_file:file create_file_perms;

# Access to /data/media.
```

```
allow anuragtrustedthree_app media_rw_data_file:dir create_dir_perms;
allow anuragtrustedthree_app media_rw_data_file:file create_file_perms;

# Write to /cache.
allow anuragtrustedthree_app cache_file:dir create_dir_perms;
allow anuragtrustedthree_app cache_file:file create_file_perms;

# Direct access to vold-mounted storage under /mnt/media_rw
# This is a performance optimization that allows anuragtrustedthree_app apps to bypass the FUSE layer
allow anuragtrustedthree_app mnt_media_rw_file:dir r_dir_perms;
allow anuragtrustedthree_app vfat:dir create_dir_perms;
allow anuragtrustedthree_app vfat:file create_file_perms;

# com.android.systemui
allow anuragtrustedthree_app rootfs:dir getattr;

allow anuragtrustedthree_app audioserver_service:service_manager find;
allow anuragtrustedthree_app cameraserver_service:service_manager find;
allow anuragtrustedthree_app drmserver_service:service_manager find;
allow anuragtrustedthree_app mediaserver_service:service_manager find;
allow anuragtrustedthree_app mediametrics_service:service_manager find;
allow anuragtrustedthree_app mediaextractor_service:service_manager find;
allow anuragtrustedthree_app mediacodec_service:service_manager find;
allow anuragtrustedthree_app mediadrmserver_service:service_manager find;
allow anuragtrustedthree_app persistent_data_block_service:service_manager find;
allow anuragtrustedthree_app radio_service:service_manager find;
allow anuragtrustedthree_app surfaceflinger_service:service_manager find;
allow anuragtrustedthree_app timezone_service:service_manager find;
allow anuragtrustedthree_app app_api_service:service_manager find;
allow anuragtrustedthree_app system_api_service:service_manager find;
allow anuragtrustedthree_app vr_manager_service:service_manager find;

# Access to /data/preloads
allow anuragtrustedthree_app preloads_data_file:file r_file_perms;
allow anuragtrustedthree_app preloads_data_file:dir r_dir_perms;
allow anuragtrustedthree_app preloads_media_file:file r_file_perms;
allow anuragtrustedthree_app preloads_media_file:dir r_dir_perms;

###
### Neverallow rules
###

# app domains which access /dev/fuse should not run as anuragtrustedthree_app
neverallow anuragtrustedthree_app fuse_device:chr_file *;
```

The Below following lines getting added to `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/anuragtrusted_app` by taking `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/platform_app` as refernece.

```
###
### Apps signed with the anuragtrusted_app key.
###
type anuragtrusted_app, domain;
typeattribute anuragtrusted_app coredomain;


app_domain(anuragtrusted_app)

# Access the network.
net_domain(anuragtrusted_app)
# Access bluetooth.
bluetooth_domain(anuragtrusted_app)
# Read from /data/local/tmp or /data/data/com.android.shell.
allow anuragtrusted_app shell_data_file:dir search;
allow anuragtrusted_app shell_data_file:file { open getattr read };
allow anuragtrusted_app icon_file:file { open getattr read };

# ASEC
allow anuragtrusted_app asec_apk_file:dir create_dir_perms;
allow anuragtrusted_app asec_apk_file:file create_file_perms;

# Access to /data/media.
allow anuragtrusted_app media_rw_data_file:dir create_dir_perms;
allow anuragtrusted_app media_rw_data_file:file create_file_perms;

# Write to /cache.
allow anuragtrusted_app cache_file:dir create_dir_perms;
allow anuragtrusted_app cache_file:file create_file_perms;

# Direct access to vold-mounted storage under /mnt/media_rw
# This is a performance optimization that allows anuragtrusted_app apps to bypass the FUSE layer
allow anuragtrusted_app mnt_media_rw_file:dir r_dir_perms;
allow anuragtrusted_app vfat:dir create_dir_perms;
allow anuragtrusted_app vfat:file create_file_perms;

allow anuragtrusted_app audioserver_service:service_manager find;
allow anuragtrusted_app cameraserver_service:service_manager find;
allow anuragtrusted_app drmserver_service:service_manager find;
allow anuragtrusted_app mediaserver_service:service_manager find;
allow anuragtrusted_app mediametrics_service:service_manager find;
allow anuragtrusted_app mediaextractor_service:service_manager find;
allow anuragtrusted_app mediacodec_service:service_manager find;
allow anuragtrusted_app mediadrmserver_service:service_manager find;
allow anuragtrusted_app mediacasserver_service:service_manager find;
allow anuragtrusted_app persistent_data_block_service:service_manager find;
allow anuragtrusted_app radio_service:service_manager find;
allow anuragtrusted_app surfaceflinger_service:service_manager find;
allow anuragtrusted_app app_api_service:service_manager find;
allow anuragtrusted_app system_api_service:service_manager find;
allow anuragtrusted_app vr_manager_service:service_manager find;

# Access to /data/preloads
allow anuragtrusted_app preloads_data_file:file r_file_perms;
allow anuragtrusted_app preloads_data_file:dir r_dir_perms;
allow anuragtrusted_app preloads_media_file:file r_file_perms;
allow anuragtrusted_app preloads_media_file:dir r_dir_perms;


###
### Neverallow rules
###

# app domains which access /dev/fuse should not run as anuragtrusted_app
neverallow anuragtrusted_app fuse_device:chr_file *;
```

The Below following lines getting added to `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/anuragtrustedtwo_app` by taking `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/platform_app` as refernece.

```
###
### Apps signed with the anuragtrustedtwo_app key.
###
type anuragtrustedtwo_app, domain;
typeattribute anuragtrustedtwo_app coredomain;


app_domain(anuragtrustedtwo_app)

# Access the network.
net_domain(anuragtrustedtwo_app)
# Access bluetooth.
bluetooth_domain(anuragtrustedtwo_app)
# Read from /data/local/tmp or /data/data/com.android.shell.
allow anuragtrustedtwo_app shell_data_file:dir search;
allow anuragtrustedtwo_app shell_data_file:file { open getattr read };
allow anuragtrustedtwo_app icon_file:file { open getattr read };

# ASEC
allow anuragtrustedtwo_app asec_apk_file:dir create_dir_perms;
allow anuragtrustedtwo_app asec_apk_file:file create_file_perms;

# Access to /data/media.
allow anuragtrustedtwo_app media_rw_data_file:dir create_dir_perms;
allow anuragtrustedtwo_app media_rw_data_file:file create_file_perms;

# Write to /cache.
allow anuragtrustedtwo_app cache_file:dir create_dir_perms;
allow anuragtrustedtwo_app cache_file:file create_file_perms;

# Direct access to vold-mounted storage under /mnt/media_rw
# This is a performance optimization that allows anuragtrustedtwo_app apps to bypass the FUSE layer
allow anuragtrustedtwo_app mnt_media_rw_file:dir r_dir_perms;
allow anuragtrustedtwo_app vfat:dir create_dir_perms;
allow anuragtrustedtwo_app vfat:file create_file_perms;

allow anuragtrustedtwo_app audioserver_service:service_manager find;
allow anuragtrustedtwo_app cameraserver_service:service_manager find;
allow anuragtrustedtwo_app drmserver_service:service_manager find;
allow anuragtrustedtwo_app mediaserver_service:service_manager find;
allow anuragtrustedtwo_app mediametrics_service:service_manager find;
allow anuragtrustedtwo_app mediaextractor_service:service_manager find;
allow anuragtrustedtwo_app mediacodec_service:service_manager find;
allow anuragtrustedtwo_app mediadrmserver_service:service_manager find;
allow anuragtrustedtwo_app mediacasserver_service:service_manager find;
allow anuragtrustedtwo_app persistent_data_block_service:service_manager find;
allow anuragtrustedtwo_app radio_service:service_manager find;
allow anuragtrustedtwo_app surfaceflinger_service:service_manager find;
allow anuragtrustedtwo_app app_api_service:service_manager find;
allow anuragtrustedtwo_app system_api_service:service_manager find;
allow anuragtrustedtwo_app vr_manager_service:service_manager find;

# Access to /data/preloads
allow anuragtrustedtwo_app preloads_data_file:file r_file_perms;
allow anuragtrustedtwo_app preloads_data_file:dir r_dir_perms;
allow anuragtrustedtwo_app preloads_media_file:file r_file_perms;
allow anuragtrustedtwo_app preloads_media_file:dir r_dir_perms;


###
### Neverallow rules
###

# app domains which access /dev/fuse should not run as anuragtrustedtwo_app
neverallow anuragtrustedtwo_app fuse_device:chr_file *;
```

The Below following lines getting added to `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/anuragtrustedthree_app` by taking `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/platform_app` as refernece.

```
###
### Apps signed with the anuragtrustedthree_app key.
###
type anuragtrustedthree_app, domain;
typeattribute anuragtrustedthree_app coredomain;


app_domain(anuragtrustedthree_app)

# Access the network.
net_domain(anuragtrustedthree_app)
# Access bluetooth.
bluetooth_domain(anuragtrustedthree_app)
# Read from /data/local/tmp or /data/data/com.android.shell.
allow anuragtrustedthree_app shell_data_file:dir search;
allow anuragtrustedthree_app shell_data_file:file { open getattr read };
allow anuragtrustedthree_app icon_file:file { open getattr read };

# ASEC
allow anuragtrustedthree_app asec_apk_file:dir create_dir_perms;
allow anuragtrustedthree_app asec_apk_file:file create_file_perms;

# Access to /data/media.
allow anuragtrustedthree_app media_rw_data_file:dir create_dir_perms;
allow anuragtrustedthree_app media_rw_data_file:file create_file_perms;

# Write to /cache.
allow anuragtrustedthree_app cache_file:dir create_dir_perms;
allow anuragtrustedthree_app cache_file:file create_file_perms;

# Direct access to vold-mounted storage under /mnt/media_rw
# This is a performance optimization that allows anuragtrustedthree_app apps to bypass the FUSE layer
allow anuragtrustedthree_app mnt_media_rw_file:dir r_dir_perms;
allow anuragtrustedthree_app vfat:dir create_dir_perms;
allow anuragtrustedthree_app vfat:file create_file_perms;

allow anuragtrustedthree_app audioserver_service:service_manager find;
allow anuragtrustedthree_app cameraserver_service:service_manager find;
allow anuragtrustedthree_app drmserver_service:service_manager find;
allow anuragtrustedthree_app mediaserver_service:service_manager find;
allow anuragtrustedthree_app mediametrics_service:service_manager find;
allow anuragtrustedthree_app mediaextractor_service:service_manager find;
allow anuragtrustedthree_app mediacodec_service:service_manager find;
allow anuragtrustedthree_app mediadrmserver_service:service_manager find;
allow anuragtrustedthree_app mediacasserver_service:service_manager find;
allow anuragtrustedthree_app persistent_data_block_service:service_manager find;
allow anuragtrustedthree_app radio_service:service_manager find;
allow anuragtrustedthree_app surfaceflinger_service:service_manager find;
allow anuragtrustedthree_app app_api_service:service_manager find;
allow anuragtrustedthree_app system_api_service:service_manager find;
allow anuragtrustedthree_app vr_manager_service:service_manager find;

# Access to /data/preloads
allow anuragtrustedthree_app preloads_data_file:file r_file_perms;
allow anuragtrustedthree_app preloads_data_file:dir r_dir_perms;
allow anuragtrustedthree_app preloads_media_file:file r_file_perms;
allow anuragtrustedthree_app preloads_media_file:dir r_dir_perms;



###
### Neverallow rules
###

# app domains which access /dev/fuse should not run as anuragtrustedthree_app
neverallow anuragtrustedthree_app fuse_device:chr_file *;
```

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with SE Android domains.

Chapter 7

# Integration of Whitelisted user applications as Stock applications

## 7.1 Objective

We want to allow only set of Whitelisted user applications only as part of Customized Android Platform.

## 7.2 Integration of Whitelisted Android Application as Stock application

In previous chapter we had successfully created three domains namely `anuragtrusted_app` ,`anuragtrustedtwo_app` and `anuragtrustedthree_app`. In this section we want to incorporate three Bluetooth applications under different domains.

### 7.2.1 Creation of Application Folders

We need to create three folders for three applications under the path `AOSP_Root/packages/apps/`.In this example we had created three folders with names `Bluetoothanuragtrusted` , `Bluetoothanuragtrustedtwo` and `Bluetoothanuragtrustedthree` Respectively.

### 7.2.2 Copying of Precompiled Whitelisted Application to folders

We are already having three precompiled Bluetooth application with naming conventions `Bluetoothanuragtrusted.apk` , `Bluetoothanuragtrustedtwo.apk` and `Bluetoothanuragtrustedthree.apk` copy the each application to corresponding application foders `Bluetoothanuragtrusted` , `Bluetoothanuragtrustedtwo` and `Bluetoothanuragtrustedthree` which are located in the following path `AOSP_Root/packages/apps/`

### 7.2.3 Creation of Android.mk file corresponding to each application

Now create three `Android.mk` files in three folders `Bluetoothanuragtrusted` , `Bluetoothanuragtrustedtwo` and `Bluetoothanuragtrustedthree`
The following lines need to be included in the `AOSP_Root/packages/apps/Bluetoothanuragtrusted/Android.mk`

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
# Module name should match apk name to be installed.
LOCAL_MODULE := Bluetoothanuragtrusted
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := anurag
include $(BUILD_PREBUILT)
```

The following lines need to be included in the `AOSP_Root/packages/apps/Bluetoothanuragtrustedtwo/Android.mk`

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
# Module name should match apk name to be installed.
LOCAL_MODULE := Bluetoothanuragtrustedtwo
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := anuragtwo
include $(BUILD_PREBUILT)
```

The following lines need to be included in the `AOSP_Root/packages/apps/Bluetoothanuragtrustedthree/Android.mk`

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
# Module name should match apk name to be installed.
LOCAL_MODULE := Bluetoothanuragtrustedthree
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := anuragthree
include $(BUILD_PREBUILT)
```

### 7.2.4 Integration of Application folders in Build process

Identify the following file `AOSP_Root/build/target/product/core.mk` . The following lines before inclusion of Bluetooth application folders

```
#
# Copyright (C) 2007 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Base configuration for communication-oriented android devices
# (phones, tablets, etc.).  If you want a change to apply to ALMOST ALL
# devices (including non-phones and non-tablets), modify
# core_minimal.mk instead. If you care about wearables, you need to modify
# core_tiny.mk in addition to core_minimal.mk.

PRODUCT_PACKAGES += \
    BasicDreams \
    BlockedNumberProvider \
    BookmarkProvider \
    BuiltInPrintService \
    CalendarProvider \
    CaptivePortalLogin \
    CertInstaller \
    DownloadProviderUi \
    ExternalStorageProvider \
    FusedLocation \
    InputDevices \
    KeyChain \
    Keyguard \
    LatinIME \
    Launcher2 \
    ManagedProvisioning \
    MtpDocumentsProvider \
    PicoTts \
    PacProcessor \
    libpac \
    PrintSpooler \
    PrintRecommendationService \
    ProxyHandler \
    Settings \
    SharedStorageBackup \
    StorageManager \
    Telecom \
    TeleService \
    VpnDialogs \
    vr \
    MmsService

# The set of packages whose code can be loaded by the system server.
PRODUCT_SYSTEM_SERVER_APPS += \
    FusedLocation \
    InputDevices \
    KeyChain \
    Telecom \

# The set of packages we want to force 'speed' compilation on.
PRODUCT_DEXPREOPT_SPEED_APPS += \

$(call inherit-product, $(SRC_TARGET_DIR)/product/core_base.mk)
```

The following lines after inclusion of Bluetooth application folders

```
#
# Copyright (C) 2007 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# Base configuration for communication-oriented android devices
# (phones, tablets, etc.).  If you want a change to apply to ALMOST ALL
# devices (including non-phones and non-tablets), modify
# core_minimal.mk instead. If you care about wearables, you need to modify
# core_tiny.mk in addition to core_minimal.mk.

PRODUCT_PACKAGES += \
    BasicDreams \
    BlockedNumberProvider \
    BookmarkProvider \
    BuiltInPrintService \
    CalendarProvider \
    CaptivePortalLogin \
    CertInstaller \
    DownloadProviderUi \
    ExternalStorageProvider \
    FusedLocation \
    InputDevices \
    KeyChain \
    Keyguard \
    LatinIME \
    Launcher2 \
    ManagedProvisioning \
    MtpDocumentsProvider \
    PicoTts \
    PacProcessor \
    libpac \
    PrintSpooler \
    PrintRecommendationService \
    ProxyHandler \
    Settings \
    SharedStorageBackup \
    StorageManager \
    Telecom \
    TeleService \
    VpnDialogs \
    vr \
    MmsService \
    Bluetoothanuragtrusted \
    Bluetoothanuragtrustedtwo \
    Bluetoothanuragtrustedthree

# The set of packages whose code can be loaded by the system server.
PRODUCT_SYSTEM_SERVER_APPS += \
    FusedLocation \
    InputDevices \
    KeyChain \
    Telecom \

# The set of packages we want to force 'speed' compilation on.
PRODUCT_DEXPREOPT_SPEED_APPS += \

$(call inherit-product, $(SRC_TARGET_DIR)/product/core_base.mk)
```

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with integrated whitelisted stock applications.

# Chapter 8

# Resource Control

## 8.1 Objective

Resource control(i.e SDcard,Bluetooth,Network,Camera) is one of the key factor in any mobile device . The rapid evolution in mobile industry with updated technology has lead to the increased internal sdcard support. Nowadays major oem manufacturers ( i.e google , oneplus , etc..) are building mobiles with higher internal storage support. Google pixel , pixel 2 and oneplus mobile are live examples in existing trends in mobile industry. Above all made clear that storage is one of the critical entity in any device.so applications which are accessing the storage plays a crucial/vital role in data security. So we come with new design using SEAndroid to ensure only Trusted signed application can only have the privilege to access this critical resource

## 8.2 Restriction of SDCard access

To achieve the Restricted SDCard access, Initially we are Restricting SDCard access to all untrusted domains using SEAndroid policy. To Achieve this Restriction capabilty we need to target `app.te` file in both the directories `AOSP_Root/system/sepolicy/private` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private`

### 8.2.1 app.te

The below mentioned lines in `app.te` file need to be identified in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict sdcard card access to untrusted applications.

```
allow { appdomain -isolated_app -ephemeral_app } storage_file:dir r_dir_perms;
allow { appdomain -isolated_app -ephemeral_app } storage_file:lnk_file r_file_perms;
allow { appdomain -isolated_app -ephemeral_app } mnt_user_file:dir r_dir_perms;
allow { appdomain -isolated_app -ephemeral_app } mnt_user_file:lnk_file r_file_perms;
```

The above mentioned four lines need to be replaced with below lines in in `app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict sdcard card access to untrusted applications.

```
allow { appdomain -isolated_app -ephemeral_app -untrusted_app_all -untrusted_v2_app } storage_file:dir r_dir_perms;
allow { appdomain -isolated_app -ephemeral_app -untrusted_app_all -untrusted_v2_app } storage_file:lnk_file r_file_perms;
allow { appdomain -isolated_app -ephemeral_app -untrusted_app_all -untrusted_v2_app } mnt_user_file:dir r_dir_perms;
allow { appdomain -isolated_app -ephemeral_app -untrusted_app_all -untrusted_v2_app } mnt_user_file:lnk_file r_file_perms;
```

The below mentioned lines in `app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict Read/write operations in sdcard

```
allow { appdomain -isolated_app -ephemeral_app } fuse:dir create_dir_perms;
allow { appdomain -isolated_app -ephemeral_app } fuse:file create_file_perms;
allow { appdomain -isolated_app -ephemeral_app } sdcardfs:dir create_dir_perms;
allow { appdomain -isolated_app -ephemeral_app } sdcardfs:file create_file_perms;
```

After lines get commented in `app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow { appdomain -isolated_app -ephemeral_app } fuse:dir create_dir_perms;
#allow { appdomain -isolated_app -ephemeral_app } fuse:file create_file_perms;
#allow { appdomain -isolated_app -ephemeral_app } sdcardfs:dir create_dir_perms;
#allow { appdomain -isolated_app -ephemeral_app } sdcardfs:file create_file_perms;
```

The below mentioned lines in `app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` since normally all external sdcards uses vfat file system. Below mentioned lines are being used to read and write operations on vfat type.

```
allow { appdomain -isolated_app -ephemeral_app } vfat:dir r_dir_perms;
allow { appdomain -isolated_app -ephemeral_app } vfat:file rw_file_perms;
```

After lines get commented in `app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow { appdomain -isolated_app -ephemeral_app } vfat:dir r_dir_perms;
#allow { appdomain -isolated_app -ephemeral_app } vfat:file rw_file_perms;
```

## 8.3 Allowing SDCard access to anuragtrustedtwo_-app and anuragtrustedthree_app domains only

To achieve the Restricted SDCard access, in previous section we had successfully restricted SDCard access to all untrusted domains. In current section we are going to allow sdcard access capability ony to `anuragtrustedtwo_app` and `anuragtrustedthree_app` domains only. To achieve this level of resource control the following lines need to added at the end of the `app.te` file in both `AOSP_Root/system/sepolicy/private` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private` directories.

```
allow anuragtrustedtwo_app fuse:dir create_dir_perms;
allow anuragtrustedtwo_app fuse:file create_file_perms;
allow anuragtrustedtwo_app sdcardfs:dir create_dir_perms;
allow anuragtrustedtwo_app sdcardfs:file create_file_perms;

allow anuragtrustedtwo_app vfat:dir r_dir_perms;
allow anuragtrustedtwo_app vfat:file rw_file_perms;

allow anuragtrustedthree_app fuse:dir create_dir_perms;
allow anuragtrustedthree_app fuse:file create_file_perms;
allow anuragtrustedthree_app sdcardfs:dir create_dir_perms;
allow anuragtrustedthree_app sdcardfs:file create_file_perms;

allow anuragtrustedthree_app vfat:dir r_dir_perms;
allow anuragtrustedthree_app vfat:file rw_file_perms;
```

## 8.4 Restricted Network/Internet access

To achieve the Restricted Network access, Initially we are Restricting Network access to all untrusted domains including anuragtrustedtwo_app,anuragtrustedthree_-app but allowing only to `anuragtrusted_app` using SEAndroid policy. To Achieve this Restriction capabilty we need to target different domain policy files in both the directories `AOSP_Root/system/sepolicy/private` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private`

### 8.4.1 platform_app.te

The below mentioned line in `platform_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to platform domain applications

```
net_domain(platform_app)
```

After lines get commented in `platform_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#net_domain(platform_app)
```

### 8.4.2 priv_app.te

The below mentioned line in `priv_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to priv app domain applications

```
net_domain(priv_app)
```

After lines get commented in `priv_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#net_domain(priv_app)
```

### 8.4.3 untrusted_app.te

The below mentioned line in `untrusted_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to untrusted app domain applications

```
net_domain(untrusted_app)
```

After lines get commented in `untrusted_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
net_domain(untrusted_app)
```

### 8.4.4 untrusted_v2_app.te

The below mentioned line in `untrusted_v2_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to untrusted app domain applications

```
net_domain(untrusted_v2_app)
```

After lines get commented in `untrusted_v2_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#net_domain(untrusted_v2_app)
```

### 8.4.5 untrusted_app_25.te

The below mentioned line in `untrusted_app_25.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to untrusted app domain applications

```
net_domain(untrusted_app_25)
```

After lines get commented in `untrusted_app_25.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#net_domain(untrusted_app_25)
```

### 8.4.6 anuragtrustedtwo_app.te

The below mentioned line in `anuragtrustedtwo_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to untrusted app domain applications

```
net_domain(anuragtrustedtwo_app)
```

After lines get commented in `anuragtrustedtwo_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#net_domain(anuragtrustedtwo_app)
```

### 8.4.7 anuragtrustedthree_app.te

The below mentioned line in `anuragtrustedthree_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to untrusted app domain applications

```
net_domain(anuragtrustedthree_app)
```

After lines get commented in `anuragtrustedthree_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#net_domain(anuragtrustedthree_app)
```

## 8.5 Restricted Cammera access

To achieve the Restricted Camera access, Initially we are Restricting Camera service access to all untrusted domains including anuragtrusted_app,anuragtrustedthree_app but allowing only to `anuragtrustedtwo_app` using SEAndroid policy. To Achieve this Restriction capabilty we need to target different domain policy files in both the directories `AOSP_Root/system/sepolicy/private` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private`

### 8.5.1 platform_app.te

The below mentioned line in `platform_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict Camera service access to platform domain applications

```
allow platform_app cameraserver_service:service_manager find;
```

After lines get commented in `platform_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow platform_app cameraserver_service:service_manager find;
```

### 8.5.2 priv_app.te

The below mentioned line in `priv_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict Camera service access to priv app domain applications

```
allow priv_app cameraserver_service:service_manager find;
```

After lines get commented in `priv_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow priv_app cameraserver_service:service_manager find;
```

### 8.5.3 untrusted_app_all.te

The below mentioned line in `untrusted_app_all.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict Camera service access to untrusted app domain applications

```
allow untrusted_app_all cameraserver_service:service_manager find;
```

After lines get commented in `untrusted_app_all.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow untrusted_app_all cameraserver_service:service_manager find;
```

### 8.5.4 untrusted_v2_app.te

The below mentioned line in `untrusted_v2_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict network access to untrusted app domain applications

```
allow untrusted_v2_app cameraserver_service:service_manager find;
```

After lines get commented in `untrusted_v2_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow untrusted_v2_app cameraserver_service:service_manager find;
```

### 8.5.5 anuragtrusted_app.te

The below mentioned line in `anuragtrusted_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict Camera service access to untrusted app domain applications

```
allow anuragtrusted_app cameraserver_service:service_manager find;
```

After lines get commented in `anuragtrusted_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow anuragtrusted_app cameraserver_service:service_manager find;
```

### 8.5.6 anuragtrustedthree_app.te

The below mentioned line in `anuragtrustedthree_app.te` file need to be commented in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` to restrict Camera service access to untrusted app domain applications

```
allow anuragtrustedthree_app cameraserver_service:service_manager find;
```

After lines get commented in `anuragtrustedthree_app.te` file in both directories `AOSP_Root/system/sepolicy/private/` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private/` the commented lines will be as follows

```
#allow anuragtrustedthree_app cameraserver_service:service_manager find;
```

## 8.6 Restricted Bluetooth access

To achieve Restricted Bluetooth access we had followed the concept of "Privileged Permission Whitelisting".Privileged applications are system applications located in the /system/priv-app directory on the system image.Starting in Android 8.0, implementors can explicitly whitelist privileged apps in the system configuration XML files in the /etc/permissions directory. Apps not explicitly listed in these XML files are not granted privileged permissions. To achieve Restricted Bluetooh access we are making normal `android.permission.BLUETOOTH` permission as signature/priviliged permission.

### 8.6.1 Making android.permission.BLUETOOTH as privileged permission

The below mentioned lines in `AndroidManifest.xml` file need to be identified in directory `AOSP_Root/frameworks/base/core/res/`

```
<permission android:name="android.permission.BLUETOOTH"
    android:description="@string/permdesc_bluetooth"
    android:label="@string/permlab_bluetooth"
    android:protectionLevel="normal" />
```

In the above lines the protection level of the permission need to be modified from `normal` to `signature|privileged` .Once after successfull modification the modified lines in `AndroidManifest.xml` file we be as follows

```
<permission android:name="android.permission.BLUETOOTH"
    android:description="@string/permdesc_bluetooth"
    android:label="@string/permlab_bluetooth"
    android:protectionLevel="signature|privileged" />
```

### 8.6.2 Making an application as privleged permission accessible

We had successully made android.permission.BLUETOOTH as privileged permission in above section . Now to access above permission capability in our applications or existed applications we need customize Application build Makefile. The following `LOCAL_PRIVILEGED_MODULE:=true` attribute need to be included in `Android.mk` file of the Android applications. During Application permission profiling it had been observed that Bluetooth and Nfc applications are internally using BLUETOOTH permission. So we are making the Bluetooth and Nfc applications as priviliged applications .

The following line need to be identified in `AOSP_Root/packages/apps/Bluetooth/Android.mk` file of Bluetooth application.

```
LOCAL_CERTIFICATE := platform
```

Next below the above line add the following line

```
LOCAL_PRIVILEGED_MODULE := true
```

After adding the following `LOCAL_PRIVILEGED_MODULE:=true` atribute . The modified `AOSP_Root/packages/apps/Bluetooth/Android.mk` file will be as follows

```
LOCAL_CERTIFICATE := platform
LOCAL_PRIVILEGED_MODULE := true
```

Now we are targeting Nfc application.The following line need to be identified in `AOSP_Root/packages/apps/Nfc/Android.mk` file of Nfc application.

```
LOCAL_CERTIFICATE := platform
```

Next below the above line add the following line

```
LOCAL_PRIVILEGED_MODULE := true
```

After adding the following `LOCAL_PRIVILEGED_MODULE:=true` atribute . The modified `AOSP_Root/packages/apps/Nfc/Android.mk` file will be as follows

```
LOCAL_CERTIFICATE := platform
LOCAL_PRIVILEGED_MODULE := true
```

Now we are targeting Bluetoothanuragtrustedthree application.The following line need to be identified in `AOSP_Root/packages/apps/Bluetoothanuragtrustedthree/Android.mk` file of Bluetoothanuragtrustedthree application.

```
LOCAL_CERTIFICATE := anuragthree
```

Next below the above line add the following line

```
LOCAL_PRIVILEGED_MODULE := true
```

After adding the following `LOCAL_PRIVILEGED_MODULE:=true` atribute . The modified `AOSP_Root/packages/apps/Bluetoothanuragtrustedthree/Android.mk` file will be as follows

```
LOCAL_CERTIFICATE := anuragthree
LOCAL_PRIVILEGED_MODULE := true
```

### 8.6.3 Updation of privapp-permissions-platform.xml

Before going to Updation of `AOSP_Root/frameworks/base/data/etc/privapp-permissions-platform.xml` first we need to undergo the complete framework build process and new flashable zip file need to be generated. Once after successfull generation of zip file.
Execute below command from `AOSP_Root`

```
$ python development/tools/privapp_permissions/privapp_permissions.py
```

The successfull execution of above command will lists all signature|privileged permissions required to be whitelisted specific to applications. The resultant output will be as follows.

```
<?xml version="1.0" encoding="utf-8"?>
<permissions>
    <privapp-permissions package="android.ext.services">
        <permission name="android.permission.PROVIDE_RESOLVER_RANKER_SERVICE"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.apps.tag">
        <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.bluetooth">
        <permission name="android.permission.BLUETOOTH"/>
        <permission name="android.permission.BLUETOOTH_PRIVILEGED"/>
        <permission name="android.permission.CALL_PRIVILEGED"/>
        <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
        <permission name="android.permission.DUMP"/>
        <permission name="android.permission.INTERACT_ACROSS_USERS"/>
        <permission name="android.permission.MANAGE_USERS"/>
        <permission name="android.permission.MEDIA_CONTENT_CONTROL"/>
        <permission name="android.permission.MODIFY_AUDIO_ROUTING"/>
        <permission name="android.permission.MODIFY_PHONE_STATE"/>
        <permission name="android.permission.NFC_HANDOVER_STATUS"/>
        <permission name="android.permission.READ_PRIVILEGED_PHONE_STATE"/>
        <permission name="android.permission.REAL_GET_TASKS"/>
        <permission name="android.permission.TETHER_PRIVILEGED"/>
        <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
        <permission name="android.permission.UPDATE_DEVICE_STATS"/>
        <permission name="android.permission.WRITE_APN_SETTINGS"/>
        <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.dialer">
        <permission name="android.permission.STATUS_BAR"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.musicfx">
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>
    <privapp-permissions package="com.android.nfc">
        <permission name="android.permission.BLUETOOTH"/>
        <permission name="android.permission.BLUETOOTH_PRIVILEGED"/>
        <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
        <permission name="android.permission.DISPATCH_NFC_MESSAGE"/>
        <permission name="android.permission.LOCAL_MAC_ADDRESS"/>
        <permission name="android.permission.MANAGE_USERS"/>
        <permission name="android.permission.MASTER_CLEAR"/>
        <permission name="android.permission.NFC_HANDOVER_STATUS"/>
        <permission name="android.permission.OVERRIDE_WIFI_CONFIG"/>
        <permission name="android.permission.READ_FRAME_BUFFER"/>
        <permission name="android.permission.REAL_GET_TASKS"/>
        <permission name="android.permission.STATUS_BAR"/>
        <permission name="android.permission.STOP_APP_SWITCHES"/>
        <permission name="android.permission.USER_ACTIVITY"/>
        <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.phone">
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.server.telecom">
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.settings">
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.shell">
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.systemui">
        <permission name="android.permission.BLUETOOTH"/>
        <permission name="com.android.permission.WRITE_EMBEDDED_SUBSCRIPTIONS"/>
    </privapp-permissions>

    <privapp-permissions package="com.santossingh.bft">
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

</permissions>
```

Taking above suggestions into considerations the `AOSP_Root/frameworks/base/data/etc/privapp-permissions-platform.xml` file need to updated. The updated file be as follows

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  ~ Copyright (C) 2016 The Android Open Source Project
  ~
  ~ Licensed under the Apache License, Version 2.0 (the "License");
  ~ you may not use this file except in compliance with the License.
  ~ You may obtain a copy of the License at
  ~
  ~      http://www.apache.org/licenses/LICENSE-2.0
  ~
  ~ Unless required by applicable law or agreed to in writing, software
  ~ distributed under the License is distributed on an "AS IS" BASIS,
  ~ WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  ~ See the License for the specific language governing permissions and
  ~ limitations under the License
  -->

<!--
This XML file declares which signature|privileged permissions should be granted to privileged
applications that come with the platform
-->
<permissions>
    <privapp-permissions package="com.android.backupconfirm">
        <permission name="android.permission.BACKUP"/>
        <permission name="android.permission.CRYPT_KEEPER"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.cellbroadcastreceiver">
        <permission name="android.permission.INTERACT_ACROSS_USERS"/>
        <permission name="android.permission.MANAGE_USERS"/>
        <permission name="android.permission.MODIFY_PHONE_STATE"/>
        <permission name="android.permission.READ_PRIVILEGED_PHONE_STATE"/>
        <permission name="android.permission.RECEIVE_EMERGENCY_BROADCAST"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.contacts">
        <permission name="android.permission.GET_ACCOUNTS_PRIVILEGED"/>
        <permission name="com.android.voicemail.permission.READ_VOICEMAIL"/>
    </privapp-permissions>
```

```
<privapp-permissions package="com.android.defcontainer">
    <permission name="android.permission.ACCESS_CACHE_FILESYSTEM"/>
    <permission name="android.permission.ALLOCATE_AGGRESSIVE"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.WRITE_MEDIA_STORAGE"/>
</privapp-permissions>

<privapp-permissions package="com.android.dialer">
    <permission name="android.permission.ALLOW_ANY_CODEC_FOR_PLAYBACK"/>
    <permission name="android.permission.CONTROL_INCALL_EXPERIENCE"/>
    <permission name="android.permission.GET_ACCOUNTS_PRIVILEGED"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
    <permission name="android.permission.STOP_APP_SWITCHES"/>
    <permission name="com.android.voicemail.permission.READ_VOICEMAIL"/>
    <permission name="com.android.voicemail.permission.WRITE_VOICEMAIL"/>
</privapp-permissions>

<privapp-permissions package="com.android.emergency">
    <permission name="android.permission.MANAGE_USERS"/>
</privapp-permissions>

<privapp-permissions package="com.android.externalstorage">
    <permission name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <permission name="android.permission.WRITE_MEDIA_STORAGE"/>
</privapp-permissions>

<privapp-permissions package="com.android.launcher3">
    <permission name="android.permission.BIND_APPWIDGET"/>
    <permission name="android.permission.GET_ACCOUNTS_PRIVILEGED"/>
</privapp-permissions>

<privapp-permissions package="com.android.location.fused">
    <permission name="android.permission.INSTALL_LOCATION_PROVIDER"/>
</privapp-permissions>

<privapp-permissions package="com.android.managedprovisioning">
    <permission name="android.permission.CHANGE_COMPONENT_ENABLED_STATE"/>
    <permission name="android.permission.CHANGE_CONFIGURATION"/>
    <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
    <permission name="android.permission.CRYPT_KEEPER"/>
    <permission name="android.permission.DELETE_PACKAGES"/>
    <permission name="android.permission.INSTALL_PACKAGES"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.MANAGE_DEVICE_ADMINS"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.MASTER_CLEAR"/>
    <permission name="android.permission.PERFORM_CDMA_PROVISIONING"/>
    <permission name="android.permission.SET_TIME"/>
    <permission name="android.permission.SET_TIME_ZONE"/>
    <permission name="android.permission.SHUTDOWN"/>
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
</privapp-permissions>

<privapp-permissions package="com.android.mms.service">
    <permission name="android.permission.BIND_CARRIER_MESSAGING_SERVICE"/>
    <permission name="android.permission.BIND_CARRIER_SERVICES"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
</privapp-permissions>

<privapp-permissions package="com.android.mtp">
    <permission name="android.permission.MANAGE_USB"/>
</privapp-permissions>

<privapp-permissions package="com.android.musicfx">
    <permission name="android.permission.CHANGE_COMPONENT_ENABLED_STATE"/>
    <permission name="android.permission.BLUETOOTH"/>
</privapp-permissions>

<privapp-permissions package="com.android.networkrecommendation">
    <permission name="android.permission.SCORE_NETWORKS"/>
    <permission name="android.permission.SUBSTITUTE_NOTIFICATION_APP_NAME"/>
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
</privapp-permissions>

<privapp-permissions package="com.android.omadm.service">
    <permission name="android.permission.CHANGE_CONFIGURATION"/>
    <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
    <permission name="android.permission.READ_PRIVILEGED_PHONE_STATE"/>
    <permission name="android.permission.WRITE_APN_SETTINGS"/>
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
</privapp-permissions>

<privapp-permissions package="com.android.packageinstaller">
    <permission name="android.permission.CLEAR_APP_CACHE"/>
    <permission name="android.permission.DELETE_PACKAGES"/>
    <permission name="android.permission.INSTALL_PACKAGES"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.OBSERVE_GRANT_REVOKE_PERMISSIONS"/>
    <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
</privapp-permissions>

<privapp-permissions package="com.android.phone">
    <permission name="android.permission.ACCESS_IMS_CALL_SERVICE"/>
    <permission name="android.permission.BIND_CARRIER_MESSAGING_SERVICE"/>
    <permission name="android.permission.BIND_CARRIER_SERVICES"/>
    <permission name="android.permission.BIND_IMS_SERVICE"/>
    <permission name="android.permission.BIND_VISUAL_VOICEMAIL_SERVICE"/>
    <permission name="android.permission.CALL_PRIVILEGED"/>
    <permission name="android.permission.CHANGE_COMPONENT_ENABLED_STATE"/>
    <permission name="android.permission.CHANGE_CONFIGURATION"/>
    <permission name="android.permission.CHANGE_DEVICE_IDLE_TEMP_WHITELIST"/>
    <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
    <permission name="android.permission.CONTROL_INCALL_EXPERIENCE"/>
    <permission name="android.permission.DUMP"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.LOCAL_MAC_ADDRESS"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
    <permission name="android.permission.PERFORM_CDMA_PROVISIONING"/>
    <permission name="android.permission.READ_NETWORK_USAGE_HISTORY"/>
    <permission name="android.permission.READ_PRIVILEGED_PHONE_STATE"/>
    <permission name="android.permission.READ_SEARCH_INDEXABLES"/>
    <permission name="android.permission.REBOOT"/>
    <permission name="android.permission.REGISTER_CALL_PROVIDER"/>
    <permission name="android.permission.REGISTER_SIM_SUBSCRIPTION"/>
    <permission name="android.permission.SEND_RESPOND_VIA_MESSAGE"/>
    <permission name="android.permission.SET_TIME"/>
    <permission name="android.permission.SET_TIME_ZONE"/>
    <permission name="android.permission.SHUTDOWN"/>
    <permission name="android.permission.STATUS_BAR"/>
    <permission name="android.permission.STOP_APP_SWITCHES"/>
    <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
    <permission name="android.permission.UPDATE_DEVICE_STATS"/>
    <permission name="android.permission.UPDATE_LOCK"/>
    <permission name="android.permission.WRITE_APN_SETTINGS"/>
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    <permission name="com.android.permission.WRITE_EMBEDDED_SUBSCRIPTIONS"/>
    <permission name="com.android.voicemail.permission.READ_VOICEMAIL"/>
    <permission name="com.android.voicemail.permission.WRITE_VOICEMAIL"/>
    <permission name="android.permission.BLUETOOTH"/>
</privapp-permissions>

<privapp-permissions package="com.android.providers.calendar">
    <permission name="android.permission.GET_ACCOUNTS_PRIVILEGED"/>
    <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
</privapp-permissions>

<privapp-permissions package="com.android.providers.contacts">
    <permission name="android.permission.BIND_DIRECTORY_SEARCH"/>
    <permission name="android.permission.GET_ACCOUNTS_PRIVILEGED"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
</privapp-permissions>

<privapp-permissions package="com.android.providers.downloads">
    <permission name="android.permission.ACCESS_CACHE_FILESYSTEM"/>
    <permission name="android.permission.CLEAR_APP_CACHE"/>
    <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
    <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
    <permission name="android.permission.UPDATE_DEVICE_STATS"/>
</privapp-permissions>

<privapp-permissions package="com.android.providers.media">
    <permission name="android.permission.ACCESS_MTP"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.WRITE_MEDIA_STORAGE"/>
</privapp-permissions>

<privapp-permissions package="com.android.providers.telephony">
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
</privapp-permissions>

<privapp-permissions package="com.android.provision">
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
</privapp-permissions>

<privapp-permissions package="com.android.server.telecom">
    <permission name="android.permission.BIND_CONNECTION_SERVICE"/>
    <permission name="android.permission.BIND_INCALL_SERVICE"/>
    <permission name="android.permission.CALL_PRIVILEGED"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
    <permission name="android.permission.STOP_APP_SWITCHES"/>
    <permission name="android.permission.SUBSTITUTE_NOTIFICATION_APP_NAME"/>
    <permission name="android.permission.BLUETOOTH"/>
</privapp-permissions>

<privapp-permissions package="com.android.settings">
    <permission name="android.permission.ACCESS_CHECKIN_PROPERTIES"/>
    <permission name="android.permission.ACCESS_NOTIFICATIONS"/>
    <permission name="android.permission.BACKUP"/>
    <permission name="android.permission.BATTERY_STATS"/>
    <permission name="android.permission.BLUETOOTH_PRIVILEGED"/>
    <permission name="android.permission.CHANGE_CONFIGURATION"/>
    <permission name="android.permission.DELETE_PACKAGES"/>
    <permission name="android.permission.FORCE_STOP_PACKAGES"/>
    <permission name="android.permission.MANAGE_DEVICE_ADMINS"/>
    <permission name="android.permission.MANAGE_FINGERPRINT"/>
    <permission name="android.permission.MANAGE_USB"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.MANAGE_USER_OEM_UNLOCK_STATE" />
    <permission name="android.permission.MASTER_CLEAR"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
    <permission name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <permission name="android.permission.MOVE_PACKAGE"/>
    <permission name="android.permission.OVERRIDE_WIFI_CONFIG"/>
    <permission name="android.permission.PACKAGE_USAGE_STATS"/>
    <permission name="android.permission.READ_SEARCH_INDEXABLES"/>
    <permission name="android.permission.REBOOT"/>
    <permission name="android.permission.SET_TIME"/>
    <permission name="android.permission.STATUS_BAR"/>
    <permission name="android.permission.TETHER_PRIVILEGED"/>
    <permission name="android.permission.USER_ACTIVITY"/>
    <permission name="android.permission.WRITE_APN_SETTINGS"/>
    <permission name="android.permission.WRITE_MEDIA_STORAGE"/>
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    <permission name="android.permission.BLUETOOTH"/>
</privapp-permissions>

<privapp-permissions package="com.android.sharedstoragebackup">
    <permission name="android.permission.WRITE_MEDIA_STORAGE"/>
</privapp-permissions>

<privapp-permissions package="com.android.shell">
    <permission name="android.permission.BACKUP"/>
    <permission name="android.permission.BATTERY_STATS"/>
    <permission name="android.permission.BIND_APPWIDGET"/>
    <permission name="android.permission.CHANGE_COMPONENT_ENABLED_STATE"/>
    <permission name="android.permission.CHANGE_CONFIGURATION"/>
    <permission name="android.permission.CHANGE_DEVICE_IDLE_TEMP_WHITELIST" />
    <permission name="android.permission.CHANGE_OVERLAY_PACKAGES"/>
    <permission name="android.permission.CLEAR_APP_CACHE"/>
    <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
    <permission name="android.permission.DELETE_CACHE_FILES"/>
    <permission name="android.permission.DELETE_PACKAGES"/>
    <permission name="android.permission.DUMP"/>
    <permission name="android.permission.ACTIVITY_EMBEDDING"/>
    <permission name="android.permission.FORCE_STOP_PACKAGES"/>
    <permission name="android.permission.GET_APP_OPS_STATS"/>
    <permission name="android.permission.INSTALL_LOCATION_PROVIDER"/>
    <permission name="android.permission.INSTALL_PACKAGES"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.LOCAL_MAC_ADDRESS"/>
    <permission name="android.permission.MANAGE_ACTIVITY_STACKS"/>
    <permission name="android.permission.MANAGE_DEVICE_ADMINS"/>
    <permission name="android.permission.MANAGE_USB"/>
    <permission name="android.permission.MODIFY_APPWIDGET_BIND_PERMISSIONS"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
    <permission name="android.permission.MOUNT_FORMAT_FILESYSTEMS"/>
    <permission name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
    <permission name="android.permission.MOVE_PACKAGE"/>
    <permission name="android.permission.PACKAGE_USAGE_STATS" />
    <permission name="android.permission.READ_FRAME_BUFFER"/>
    <permission name="android.permission.REAL_GET_TASKS"/>
    <permission name="android.permission.REGISTER_CALL_PROVIDER"/>
    <permission name="android.permission.REGISTER_CONNECTION_MANAGER"/>
    <permission name="android.permission.REGISTER_SIM_SUBSCRIPTION"/>
    <permission name="android.permission.RETRIEVE_WINDOW_CONTENT"/>
    <permission name="android.permission.SET_ALWAYS_FINISH"/>
    <permission name="android.permission.SET_ANIMATION_SCALE"/>
    <permission name="android.permission.SET_DEBUG_APP"/>
    <permission name="android.permission.SET_PROCESS_LIMIT"/>
    <permission name="android.permission.SIGNAL_PERSISTENT_PROCESSES"/>
    <permission name="android.permission.STOP_APP_SWITCHES"/>
    <permission name="android.permission.SUBSTITUTE_NOTIFICATION_APP_NAME"/>
    <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
    <permission name="android.permission.WRITE_MEDIA_STORAGE"/>
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    <permission name="android.permission.BLUETOOTH"/>
</privapp-permissions>

<privapp-permissions package="com.android.statementservice">
    <permission name="android.permission.INTENT_FILTER_VERIFICATION_AGENT"/>
</privapp-permissions>

<privapp-permissions package="com.android.storagemanager">
    <permission name="android.permission.DELETE_PACKAGES"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.PACKAGE_USAGE_STATS"/>
    <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
</privapp-permissions>

<privapp-permissions package="com.android.systemui">
    <permission name="android.permission.BATTERY_STATS"/>
    <permission name="android.permission.BIND_APPWIDGET"/>
    <permission name="android.permission.BLUETOOTH_PRIVILEGED"/>
    <permission name="android.permission.CHANGE_COMPONENT_ENABLED_STATE"/>
    <permission name="android.permission.CHANGE_DEVICE_IDLE_TEMP_WHITELIST"/>
    <permission name="android.permission.CHANGE_OVERLAY_PACKAGES"/>
    <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
    <permission name="android.permission.CONTROL_VPN"/>
    <permission name="android.permission.DUMP"/>
    <permission name="android.permission.GET_APP_OPS_STATS"/>
    <permission name="android.permission.INTERACT_ACROSS_USERS"/>
    <permission name="android.permission.MANAGE_ACTIVITY_STACKS"/>
    <permission name="android.permission.MANAGE_USB"/>
    <permission name="android.permission.MANAGE_USERS"/>
    <permission name="android.permission.MASTER_CLEAR"/>
    <permission name="android.permission.MEDIA_CONTENT_CONTROL"/>
    <permission name="android.permission.MODIFY_PHONE_STATE"/>
    <permission name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
```

```
        <permission name="android.permission.OVERRIDE_WIFI_CONFIG"/>
        <permission name="android.permission.READ_DREAM_STATE"/>
        <permission name="android.permission.READ_FRAME_BUFFER"/>
        <permission name="android.permission.READ_NETWORK_USAGE_HISTORY"/>
        <permission name="android.permission.READ_PRIVILEGED_PHONE_STATE"/>
        <permission name="android.permission.REAL_GET_TASKS"/>
        <permission name="android.permission.RECEIVE_MEDIA_RESOURCE_USAGE"/>
        <permission name="android.permission.START_TASKS_FROM_RECENTS"/>
        <permission name="android.permission.STATUS_BAR"/>
        <permission name="android.permission.STOP_APP_SWITCHES"/>
        <permission name="android.permission.SUBSTITUTE_NOTIFICATION_APP_NAME"/>
        <permission name="android.permission.TETHER_PRIVILEGED"/>
        <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
        <permission name="android.permission.WRITE_DREAM_STATE"/>
        <permission name="android.permission.WRITE_MEDIA_STORAGE"/>
        <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.tv">
        <permission name="android.permission.CHANGE_HDMI_CEC_ACTIVE_SOURCE"/>
        <permission name="android.permission.DVB_DEVICE"/>
        <permission name="android.permission.GLOBAL_SEARCH"/>
        <permission name="android.permission.HDMI_CEC"/>
        <permission name="android.permission.MODIFY_PARENTAL_CONTROLS"/>
        <permission name="android.permission.READ_CONTENT_RATING_SYSTEMS"/>
        <permission name="com.android.providers.tv.permission.ACCESS_ALL_EPG_DATA"/>
        <permission name="com.android.providers.tv.permission.ACCESS_WATCHED_PROGRAMS"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.vpndialogs">
        <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
        <permission name="android.permission.CONTROL_VPN"/>
    </privapp-permissions>

    <privapp-permissions package="com.google.android.ext.services">
        <permission name="android.permission.PROVIDE_RESOLVER_RANKER_SERVICE" />
    </privapp-permissions>


    <privapp-permissions package="com.android.bluetooth">
        <permission name="android.permission.BLUETOOTH"/>
        <permission name="android.permission.BLUETOOTH_PRIVILEGED"/>
        <permission name="android.permission.CALL_PRIVILEGED"/>
        <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
        <permission name="android.permission.DUMP"/>
        <permission name="android.permission.INTERACT_ACROSS_USERS"/>
        <permission name="android.permission.MANAGE_USERS"/>
        <permission name="android.permission.MEDIA_CONTENT_CONTROL"/>
        <permission name="android.permission.MODIFY_AUDIO_ROUTING"/>
        <permission name="android.permission.MODIFY_PHONE_STATE"/>
        <permission name="android.permission.NFC_HANDOVER_STATUS"/>
        <permission name="android.permission.READ_PRIVILEGED_PHONE_STATE"/>
        <permission name="android.permission.REAL_GET_TASKS"/>
        <permission name="android.permission.TETHER_PRIVILEGED"/>
        <permission name="android.permission.UPDATE_APP_OPS_STATS"/>
        <permission name="android.permission.UPDATE_DEVICE_STATS"/>
        <permission name="android.permission.WRITE_APN_SETTINGS"/>
        <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.nfc">
        <permission name="android.permission.BLUETOOTH"/>
        <permission name="android.permission.BLUETOOTH_PRIVILEGED"/>
        <permission name="android.permission.CONNECTIVITY_INTERNAL"/>
        <permission name="android.permission.DISPATCH_NFC_MESSAGE"/>
        <permission name="android.permission.LOCAL_MAC_ADDRESS"/>
        <permission name="android.permission.MANAGE_USERS"/>
        <permission name="android.permission.MASTER_CLEAR"/>
        <permission name="android.permission.NFC_HANDOVER_STATUS"/>
        <permission name="android.permission.OVERRIDE_WIFI_CONFIG"/>
        <permission name="android.permission.READ_FRAME_BUFFER"/>
        <permission name="android.permission.REAL_GET_TASKS"/>
        <permission name="android.permission.STATUS_BAR"/>
        <permission name="android.permission.STOP_APP_SWITCHES"/>
        <permission name="android.permission.USER_ACTIVITY"/>
        <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    </privapp-permissions>

    <privapp-permissions package="com.santossingh.bft">
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

</permissions>
```

### 8.6.4 Allowing SDCard access to Bluetooth domain

This is one of the requirement to ensure Bluetooth service having SDCard redability.The following lines need to added at the end of the `bluetooth.te` file in both `AOSP_Root/system/sepolicy/private` and `AOSP_Root/system/sepolicy/prebuilts/api/26.0/private` directories.

```
allow bluetooth fuse:dir create_dir_perms;
allow bluetooth fuse:file create_file_perms;
allow bluetooth sdcardfs:dir create_dir_perms;
allow bluetooth sdcardfs:file create_file_perms;

allow bluetooth vfat:dir r_dir_perms;
allow bluetooth vfat:file rw_file_perms;
```

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with customized kernel.

## 8.7 Restricted Location access

To achieve Restricted Location access we had followed the concept of "Privileged Permission Whitelisting".Privileged applications are system applications located in the /system/priv-app directory on the system image.Starting in Android 8.0, implementors can explicitly whitelist privileged apps in the system configuration XML files in the /etc/permissions directory. Apps not explicitly listed in these XML files are not granted privileged permissions. To achieve Restricted Bluetooh access we are making dangerous `android.permission.ACCESS_FINE_LOCATION`,`android.permission-group.LOCATION` permissions as signature/priviliged permission.

### 8.7.1 Making android.permission.ACCESS_FINE_LOCATION android.permission-group.LOCATION as privileged permissions

The below mentioned lines in `AndroidManifest.xml` file need to be identified in directory `AOSP_Root/frameworks/base/core/res/`

```
<permission android:name="android.permission.ACCESS_FINE_LOCATION"
    android:permissionGroup="android.permission-group.LOCATION"
    android:label="@string/permlab_accessFineLocation"
    android:description="@string/permdesc_accessFineLocation"
    android:protectionLevel="dangerous|instant" />

<!-- Allows an app to access approximate location.
     Alternatively, you might want {@link #ACCESS_FINE_LOCATION}.
     <p>Protection level: dangerous
-->
<permission android:name="android.permission.ACCESS_COARSE_LOCATION"
    android:permissionGroup="android.permission-group.LOCATION"
    android:label="@string/permlab_accessCoarseLocation"
    android:description="@string/permdesc_accessCoarseLocation"
    android:protectionLevel="dangerous|instant" />
```

In the above lines the protection level of the permission need to be modified from `dangerous|instant` to `signature|privileged` .Once after successfull modification the modified lines in `AndroidManifest.xml` file we be as follows

```
<permission android:name="android.permission.ACCESS_FINE_LOCATION"
    android:permissionGroup="android.permission-group.LOCATION"
    android:label="@string/permlab_accessFineLocation"
    android:description="@string/permdesc_accessFineLocation"
    android:protectionLevel="signature|priviliaged" />

<!-- Allows an app to access approximate location.
     Alternatively, you might want {@link #ACCESS_FINE_LOCATION}.
     <p>Protection level: dangerous
-->
<permission android:name="android.permission.ACCESS_COARSE_LOCATION"
    android:permissionGroup="android.permission-group.LOCATION"
    android:label="@string/permlab_accessCoarseLocation"
    android:description="@string/permdesc_accessCoarseLocation"
    android:protectionLevel="signature|priviliaged" />
```

### 8.7.2 Making an application as privleged permission accessible

We had successully made android.permission.ACCESS_FINE_LOCATION and android.permission-group.LOCATION as privileged permission in above section . Now to access above permission capability in our applications or existed applications we need customize Application build Makefile. The following `LOCAL_PRIVILEGED_MODULE:=true` attribute need to be included in `Android.mk` file of the Android applications. During Application permission profiling it had been observed that Dialer,fusedlocation,settings,systemui and qualcom applications are internally using Location permission. So

Now we are targeting OsmAnd application for location resource control testing . Create an Osmand folder inside `AOSP_Root/packages/apps/Osmand` and copy the apk file inside and rename apk file to Osmand.apk.

Now create a Android.mk file in above folder with following lines

```
LOCAL_PATH := $(call my-dir)
include $(CLEAR_VARS)
# Module name should match apk name to be installed.
LOCAL_MODULE := Osmand
LOCAL_SRC_FILES := $(LOCAL_MODULE).apk
LOCAL_MODULE_CLASS := APPS
LOCAL_MODULE_SUFFIX := $(COMMON_ANDROID_PACKAGE_SUFFIX)
LOCAL_CERTIFICATE := anurag
LOCAL_PRIVILEGED_MODULE := true
include $(BUILD_PREBUILT)
```

Remember to mention this folder in `AOSP_Root/build/target/product/core.mk` file to include in build process.

Build the entire AOSP and flash into the device , Now try to access location Osmand application but you wont able to access location services.

To access the location services below section need to be followed

### 8.7.3 Updation of privapp-permissions-platform.xml

Execute below command from `AOSP_Root`

```
$ python development/tools/privapp_permissions/privapp_permissions.py
```

The successfull execution of above command will lists all signature|privileged permissions required to be whitelisted specific to applications. The resultant output will be as follows.

```
<?xml version="1.0" encoding="utf-8"?>
<permissions>
    <privapp-permissions package="android.ext.services">
        <permission name="android.permission.PROVIDE_RESOLVER_RANKER_SERVICE"/>
    </privapp-permissions>

    <privapp-permissions package="com.android.apps.tag">
        <permission name="android.permission.WRITE_SECURE_SETTINGS"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.dialer">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
        <permission name="android.permission.ACCESS_FINE_LOCATION"/>
        <permission name="android.permission.STATUS_BAR"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.location.fused">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
        <permission name="android.permission.ACCESS_FINE_LOCATION"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.phone">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
    </privapp-permissions>
    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.settings">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.shell">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
        <permission name="android.permission.ACCESS_FINE_LOCATION"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.android.systemui">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
        <permission name="com.android.permission.WRITE_EMBEDDED_SUBSCRIPTIONS"/>
    </privapp-permissions>

    <!-- Additional permissions on top of privapp-permissions-platform.xml -->
    <privapp-permissions package="com.qualcomm.ltebc">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
    </privapp-permissions>

    <privapp-permissions package="net.osmand">
        <permission name="android.permission.ACCESS_COARSE_LOCATION"/>
        <permission name="android.permission.ACCESS_FINE_LOCATION"/>
        <permission name="android.permission.BLUETOOTH"/>
    </privapp-permissions>

</permissions>
```

Taking above suggestions into considerations the `AOSP_Root/frameworks/base/data/etc/privapp-permissions-platform.xml` file need to updated by focusing on location permissions which refered in above output file.

# Chapter 9

# SELinux enforcement

## 9.1 Objective

The objective is to ensure all our SELinux policies are in enforcement mode.

## 9.2 Alway Build the framework as USER Build

To achieve the SEAndroid in enforcement mode the user must build the framework in USER build mode only.In Android we had a command to query this status—the file node enforce . Reading from this file returns the status permissive(0) or enforcing(1) depending on whether we are running in permissive or enforcing mode, respectively.

### 9.2.1 Enabling Developer options

To enable Developer options on the Google Pixel 2 XL device:

- Connect your device to your computer over USB.
- In Settings, tap About phone, then tap Build number seven (7) times.
- When you see the message ”You are a developer”, tap the back button.
- Tap Developer options and enable USB debugging.

Then execute below commands

```
$ adb shell
$ getenforce
```

Above command `getenforce` which will results the `Enforcing` . This Results states that our SEAndroid in enforcement mode only.

# Chapter 10

# Developer option protection

## 10.1 Objective

Making Developer options password protected.

## 10.2 Enabling Password protection

Open the file `BuildNumberPreferenceController.java` in this path `packages/apps/Settings/src/com/android/settings/deviceinfo/` and do below mentioned changes

Identify the following line

```
import android.widget.Toast;
```

add below new line after above line

```
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.view.View;
import android.widget.EditText;
import java.security.DigestException;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

Identify the following line

```
    private static final String KEY_BUILD_NUMBER = "build_number";
```

add below new line after above line

```
    private static final String KEY_DEVELOPER_OPTION = "968d4c7d3acacd977d77008bcab6191642f040b58bf48a5b99b372d81a05f5cc";
```

Identify the following function

```
                if (!mProcessingLastDevHit) {
                    enableDevelopmentSettings();
                }
```

inside if function replace `enableDevelopmentSettings();` with below mentioned lines

```
                    AlertDialog.Builder alert = new AlertDialog.Builder(mActivity);

                    alert.setTitle("");
                    alert.setMessage("");

                    // Set an EditText view to get user input
                    final EditText input = new EditText(mActivity);
                    alert.setView(input);

                    alert.setPositiveButton("Ok", new DialogInterface.OnClickListener() {
```

```
    public void onClick(DialogInterface dialog, int whichButton) {
        String value = input.getText().toString();

        // hashing part
        String generatedPassword = null;
        try {
            MessageDigest md = MessageDigest.getInstance("SHA256");
            md.update(value.getBytes());
            byte[] mdbytes = md.digest();
            StringBuilder sb = new StringBuilder();
            for (int i = 0; i < mdbytes.length; i++) {
                sb.append(Integer.toString((mdbytes[i] & 0xff) + 0x100, 16).substring(1));
            }
            generatedPassword = sb.toString();
        } catch (NoSuchAlgorithmException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }


        // Do something with value!
        if (generatedPassword != null && generatedPassword.equals(KEY_DEVELOPER_OPTION)) {

            enableDevelopmentSettings();
        } else

        {

            mActivity.finish();
        }
    }
});

alert.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int whichButton) {
        // Canceled.
        mActivity.finish();
    }
});
AlertDialog alert2 = alert.create();
alert2.setCancelable(false);
alert2.setCanceledOnTouchOutside(false);
alert2.show();
```

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with Developer options password protected.

# Chapter 11

# Removing immutability capability from the capability set of kernel

## 11.1 Objective

## 11.2 Removing immutability capability

In order to disable this capability, comment or remove this definition from the file `KERNEL_HOME/include/uapi/linux/capability.h` Identify the below mentioned line in above mentioned file path

```
#define CAP_AUDIT_READ        37
```

Replace above line with

```
#define CAP_AUDIT_READ        9
```

Identify the below mentioned line

```
#define CAP_LAST_CAP         CAP_AUDIT_READ
```

Replace above line with

```
#define CAP_LAST_CAP         CAP_BLOCK_SUSPEND
```

With these, changes, the no. of capabilities defined in the kernel would become 36 instead of 37

Otherwise, apply patch file capability.patch to the file KERNEL_HOME/include/uapi/linux/capability.h There are various files in the kernel source code which are using this definition and in order to avoid kernel source code compilation errors, remove the usage of this capability definition in all these files.

Identify the below mentioned line in the following path `KERNEL_HOME/include/linux/capability.h`

```
# define CAP_FS_SET       ((kernel_cap_t){{ CAP_FS_MASK_B0 \
                    | CAP_TO_MASK(CAP_LINUX_IMMUTABLE), \
                    CAP_FS_MASK_B1 } })
```

Replace above lines with

```
# define CAP_FS_SET       ((kernel_cap_t){{ CAP_FS_MASK_B0, \
                    CAP_FS_MASK_B1 } })
```

Otherwise, apply patch file linux_capability.h to the file KERNEL_HOME/include/linux/capability.h

Identify the below mentioned line in the following path `KERNEL_HOME/fs/hfsplus/ioctl.c`

```
if ((flags & (FS_IMMUTABLE_FL|FS_APPEND_FL)) ||
        inode->i_flags & (S_IMMUTABLE|S_APPEND)) {
               if (!capable(CAP_LINUX_IMMUTABLE)) {
            err = -EPERM;
            goto out_unlock_inode;
        }
}
```

Replace above lines with

```
if ((flags & (FS_IMMUTABLE_FL|FS_APPEND_FL)) ||
        inode->i_flags & (S_IMMUTABLE|S_APPEND)) {
                    err = -EPERM;
                goto out_unlock_inode;
       }
```

Otherwise, apply patch file hfsplus_ioctl.patch to KERNEL_HOME/fs/hfsplus/ioctl.c

Identify the below mentioned line in the following path `KERNEL_HOME/fs/jfs/ioctl.c`

```
if ((oldflags & JFS_IMMUTABLE_FL) ||
            ((flags ^ oldflags) &
            (JFS_APPEND_FL | JFS_IMMUTABLE_FL))) {
             if (!capable(CAP_LINUX_IMMUTABLE)) {
                mutex_unlock(&inode->i_mutex);
                err = -EPERM;
                goto setflags_out;
            }
        }
```

Replace above lines with

```
if ((oldflags & JFS_IMMUTABLE_FL) ||
            ((flags ^ oldflags) &
            (JFS_APPEND_FL | JFS_IMMUTABLE_FL))) {
                 mutex_unlock(&inode->i_mutex);
                err = -EPERM;
                goto setflags_out;
        }
```

Otherwise, apply the patch jfs_ioctl.patch to the file KERNEL_HOME/fs/jfs/ioctl.c

Identify the below mentioned line in the following path `KERNEL_HOME/fs/reiserfs/ioctl.c`

```
if (((flags ^ REISERFS_I(inode)->
                  i_attrs) & (REISERFS_IMMUTABLE_FL |
                      REISERFS_APPEND_FL))
                && !capable(CAP_LINUX_IMMUTABLE)) {
                err = -EPERM;
                goto setflags_out;
            }
```

Replace above lines with

```
if (((flags ^ REISERFS_I(inode)->
                  i_attrs) & (REISERFS_IMMUTABLE_FL |
                      REISERFS_APPEND_FL))
                 {
                err = -EPERM;
                goto setflags_out;
                 }
```

Otherwise, apply the patch reiserfs_ioctl.patch to the file KERNEL_HOME/fs/reiserfs/ioctl.c

Identify the below mentioned line in the following path `KERNEL_HOME/fs/fat/file.c`

```
if (sbi->options.sys_immutable &&
        ((attr | oldattr) & ATTR_SYS) &&
        !capable(CAP_LINUX_IMMUTABLE)) {
        err = -EPERM;
        goto out_unlock_inode;
        }
```

Replace above lines with

```
if (sbi->options.sys_immutable &&
        ((attr | oldattr) & ATTR_SYS))
         {
        err = -EPERM;
        goto out_unlock_inode;
        }
```

Otherwise, apply the patch fat_file.patch to the file KERNEL_HOME/fs/fat/file.c

Identify the below mentioned line in the following path `KERNEL_HOME/fs/ext4/ioctl.c`

```
if ((flags ^ oldflags) & (EXT4_APPEND_FL | EXT4_IMMUTABLE_FL)) {
             if (!capable(CAP_LINUX_IMMUTABLE)){
                goto flags_out;
            }
        }
```

Replace above lines with

```
if ((flags ^ oldflags) & (EXT4_APPEND_FL | EXT4_IMMUTABLE_FL)) {
           goto flags_out;
        }
```

Otherwise, apply the patch ext4_ioctl.patch to the file KERNEL_HOME/fs/ext4/ioctl.c

Identify the below mentioned line in the following path `KERNEL_HOME/fs/f2fs/file.c`

```
if ((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL)) {
        if (!capable(CAP_LINUX_IMMUTABLE)) {
            mutex_unlock(&inode->i_mutex);
            ret = -EPERM;
            goto out;
        }
    }
```

Replace above lines with

```
if ((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL)) {
            mutex_unlock(&inode->i_mutex);
            ret = -EPERM;
            goto out;
    }
```

Otherwise, apply the patch f2fs_file.patch to the file KERNEL_HOME/fs/f2fs/file.c

Identify the below mentioned line in the following path `KERNEL_HOME/fs/ocfs2/ioctl.c`

```
if ((oldflags & OCFS2_IMMUTABLE_FL) || ((flags ^ oldflags) &
        (OCFS2_APPEND_FL | OCFS2_IMMUTABLE_FL))) {
            if (!capable(CAP_LINUX_IMMUTABLE))
                goto bail_unlock;
        }
```

Replace above lines with

```
if ((oldflags & OCFS2_IMMUTABLE_FL) || ((flags ^ oldflags) &
        (OCFS2_APPEND_FL | OCFS2_IMMUTABLE_FL))) {
             goto bail_unlock;
       }
```

Otherwise, apply the patch ocfs2_ioctl.patch to the file KERNEL_HOME/fs/ocfs2/ioctl.c
Identify the below mentioned line in the following path `KERNEL_HOME/fs/gfs2/file.c`

```
if (((new_flags ^ flags) & GFS2_DIF_IMMUTABLE) &&
        !capable(CAP_LINUX_IMMUTABLE))
        goto out;
```

Replace above lines with

```
if (((new_flags ^ flags) & GFS2_DIF_IMMUTABLE)
        goto out;
```

Otherwise, apply the patch gfs2_file.patch to the file KERNEL_HOME/fs/gfs2/file.c
Identify the below mentioned line in the following path `KERNEL_HOME/fs/nilfs2/ioctl.c`

```
if (((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL)) &&
        !capable(CAP_LINUX_IMMUTABLE))
        goto out;
```

Replace above lines with

```
if (((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL))
        goto out;
```

Otherwise, apply the patch nilfs2_ioctl.patch to the file KERNEL_HOME/fs/nilfs2/ioctl.c
Identify the below mentioned line in the following path `KERNEL_HOME/fs/xfs/xfs_ioctl.c`

```
if (((ip->i_d.di_flags & (XFS_DIFLAG_IMMUTABLE | XFS_DIFLAG_APPEND)) ||
      (fa->fsx_xflags & (XFS_XFLAG_IMMUTABLE | XFS_XFLAG_APPEND))) &&
     !capable(CAP_LINUX_IMMUTABLE))
     return -EPERM;
```

Replace above lines with

```
if ((ip->i_d.di_flags &
     (XFS_DIFLAG_IMMUTABLE|XFS_DIFLAG_APPEND) ||
       (fa->fsx_xflags &
     (XFS_XFLAG_IMMUTABLE | XFS_XFLAG_APPEND)))

     return -EPERM;
```

Otherwise, apply the patch xfs_ioctl.patch to the file KERNEL_HOME/fs/xfs/xfs_-ioctl.c
Identify the below mentioned line in the following path `KERNEL_HOME/fs/btrfs/ioctl.c`

```
if ((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL)) {
        if (!capable(CAP_LINUX_IMMUTABLE)) {
            ret = -EPERM;
            goto out_unlock;
            }
        }
```

Replace above lines with

```
if ((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL)) {
            ret = -EPERM;
            goto out_unlock;
        }
```

Otherwise, apply the patch btrfs_ioctl.patch to the file KERNEL_HOME/fs/btrfs/ioctl.c

Identify the below mentioned line in the following path `KERNEL_HOME/fs/ext2/ioctl.c`

```
if ((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL)) {
        if (!capable(CAP_LINUX_IMMUTABLE)) {
            ret = -EPERM;
            goto out_unlock;
            }
        }
```

Replace above lines with

```
if ((flags ^ oldflags) & (FS_APPEND_FL | FS_IMMUTABLE_FL)) {
            ret = -EPERM;
            goto out_unlock;
        }
```

Otherwise, apply the patch ext2_ioctl.patch to the file KERNEL_HOME/fs/ext2/ioctl.c

Perform the subsection 2.1.4 in Chapter 2 Android kernel compilation and Integrating to AOSP to achieve new android operating system with customized kernel.

# Chapter 12

# Denying user space processes to write to /dev/mem or /dev/kmem

## 12.1 Objective

The requirement is to disable access to /dev/mem and /dev/kmem for user space processes. But they are disabled by default in Android kernel.

## 12.2 Verification Procedure

In order to check the same in the source code, open the file `KERNEL_HOME/arch/arm64/configs/wahoo_defconfig` and search for configuration as below
`#CONFIG_DEVMEM` is not set
`#CONFIG_DEVKMEM` is not set

Also, one can check the access to **/dev/mem** and **/dev/kmem** from the shell of the Android device with root access. No entries with name mem or kmem are found under /dev.

**Note:** In order to get root access on the device, Android framework need to be build with userdebug mode

So, in order to demonstrate that access to mem and kmem are actually disabled, enable them in the kernel configuration file **wahoo_defconfig** which is in the path **KERNEL_HOME/arch/arm64/configs** as below

Initially the configuration would be
`#CONFIG_DEVMEM` is not set
`#CONFIG_DEVKMEM` is not set

Enable them in the configuration file as below
**CONFIG_DEVMEM=y**
**CONFIG_DEVKMEM=y**

Perform the subsection 2.1.4 in Chapter 2 Android kernel compilation and Integrating to AOSP to achieve new android operating system with customized kernel with mem and kmem under /dev.

After flashing the newly generated image, check for entries of mem and kmem under /dev from the shell of the Android device with root access. This time, one can find the entries of mem and kmem under /dev.

In spite of the entries being visible under /dev, one will not be able to access any data from mem and kmem

# Chapter 13

# SMS Control

## 13.1 Objective

Restricting the number of sms a user can send per day.

## 13.2 SMS Control implementation

In order to achieve sms control mechanism the customization of framework is required . Identify the following file `AOSP_Root/frameworks/opt/telephony/src/java/com/android/internal/telephony/SmsUsageMonitor.java` Identify the below mentioned line in above mentioned file path

```
import java.util.regex.Pattern;
```

add below lines after above line

```
import android.util.Log;
import java.time.format.DateTimeFormatter;
import java.time.LocalDateTime;
```

Identify the below mentioned line

```
    private static final int DEFAULT_SMS_CHECK_PERIOD = 60000;      // 1 minute
```

replace above line with below line

```
    private static final int DEFAULT_SMS_CHECK_PERIOD = 0;
```

Identify the below mentioned line

```
private static final String ATTR_PACKAGE_SMS_POLICY = "sms-policy";
```

add below lines after above line

```
private static final String SMS_MAXLIMIT_POLICY_FILE_NAME = "maxlimit_sms_policy.xml";
private AtomicFile mLimitPolicyFile;
private static final String TAG_SMS_LIMIT_POLICY_BODY = "sms-limit-policy";
private static final String TAG_SMSLIMIT = "smslimit";
private static final String ATTR_DATE_INFO = "date";
private static final String ATTR_SMSLIMIT_SMS_COUNT = "sms-remaining-count";
```

Identify the below mentioned line

```
        mSettingsObserverHandler = new SettingsObserverHandler(mContext, mCheckEnabled);
```

add below lines after above line

```
        initializelimit();
```

Identify the below mentioned function

```
    private void removeExpiredTimestamps() {
        long beginCheckPeriod = System.currentTimeMillis() - mCheckPeriod;

        synchronized (mSmsStamp) {
            Iterator<Map.Entry<String, ArrayList<Long>>> iter = mSmsStamp.entrySet().itera
            while (iter.hasNext()) {
                Map.Entry<String, ArrayList<Long>> entry = iter.next();
                ArrayList<Long> oldList = entry.getValue();
                if (oldList.isEmpty() || oldList.get(oldList.size() - 1) < beginCheckPerio
                    iter.remove();
                }
            }
        }
    }
```

add below new funtions after above functions

```
    private void initializelimit() {

        File dir = new File(SMS_POLICY_FILE_DIRECTORY);
        mLimitPolicyFile = new AtomicFile(new File(dir, SMS_MAXLIMIT_POLICY_FILE_NAME));
        FileOutputStream limitoutfile = null;
        FileInputStream limitinfile = null;
        try {
            limitinfile = mLimitPolicyFile.openRead();
            final XmlPullParser parser = Xml.newPullParser();
            parser.setInput(limitinfile, StandardCharsets.UTF_8.name());

            XmlUtils.beginDocument(parser, TAG_SMS_LIMIT_POLICY_BODY);

            XmlUtils.nextElement(parser);
            String element = parser.getName();
            if (element.equals(TAG_SMSLIMIT)) {
                String packageName = parser.getAttributeValue(null, ATTR_DATE_INFO);
                String count = parser.getAttributeValue(null, ATTR_SMSLIMIT_SMS_COUNT);
            }
        } catch (FileNotFoundException e) {
            // No data yet
            try {
                limitoutfile = mLimitPolicyFile.startWrite();
                XmlSerializer out = new FastXmlSerializer();
                out.setOutput(limitoutfile, StandardCharsets.UTF_8.name());
                out.startDocument(null, true);
                out.startTag(null, TAG_SMS_LIMIT_POLICY_BODY);
                out.startTag(null, TAG_SMSLIMIT);
                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
                LocalDateTime now = LocalDateTime.now();
                out.attribute(null, ATTR_DATE_INFO, dtf.format(now));
                out.attribute(null, ATTR_SMSLIMIT_SMS_COUNT, "10");
                out.endTag(null, TAG_SMSLIMIT);
                out.endTag(null, TAG_SMS_LIMIT_POLICY_BODY);
                out.endDocument();

                mLimitPolicyFile.finishWrite(limitoutfile);
            } catch (IOException ex) {
                Rlog.e(TAG, "Unable to write premium SMS policy database", ex);
                if (limitoutfile != null) {
                    mLimitPolicyFile.failWrite(limitoutfile);
                }
            }
        } catch (IOException e) {
            Rlog.e(TAG, "Unable to read premium SMS policy database", e);
        } catch (NumberFormatException e) {
            Rlog.e(TAG, "Unable to parse premium SMS policy database", e);
        } catch (XmlPullParserException e) {
            Rlog.e(TAG, "Unable to parse premium SMS policy database", e);
        } finally {
            if (limitinfile != null) {
                try {
                    limitinfile.close();
                } catch (IOException ignored) {}
            }
        }

    }



    private int remainingsmscount() {
        File dir = new File(SMS_POLICY_FILE_DIRECTORY);
        mLimitPolicyFile = new AtomicFile(new File(dir, SMS_MAXLIMIT_POLICY_FILE_NAME));
        FileInputStream limitinfile = null;
        String count = "0";
        try {
            limitinfile = mLimitPolicyFile.openRead();
            final XmlPullParser parser = Xml.newPullParser();
            parser.setInput(limitinfile, StandardCharsets.UTF_8.name());
```

```
            XmlUtils.beginDocument(parser, TAG_SMS_LIMIT_POLICY_BODY);

            XmlUtils.nextElement(parser);
            String element = parser.getName();
            if (element.equals(TAG_SMSLIMIT)) {
                String packageName = parser.getAttributeValue(null, ATTR_DATE_INFO);
                count = parser.getAttributeValue(null, ATTR_SMSLIMIT_SMS_COUNT);
            }
        } catch (FileNotFoundException e) {

        } catch (IOException e) {
            Rlog.e(TAG, "Unable to read premium SMS policy database", e);
        } catch (NumberFormatException e) {
            Rlog.e(TAG, "Unable to parse premium SMS policy database", e);
        } catch (XmlPullParserException e) {
            Rlog.e(TAG, "Unable to parse premium SMS policy database", e);
        } finally {
            if (limitinfile != null) {
                try {
                    limitinfile.close();
                } catch (IOException ignored) {}
            }
        }
        return Integer.parseInt(count);
    }

    private void updatecount(int latestcount) {

        File dir = new File(SMS_POLICY_FILE_DIRECTORY);
        mLimitPolicyFile = new AtomicFile(new File(dir, SMS_MAXLIMIT_POLICY_FILE_NAME));
        FileOutputStream limitoutfile = null;

        // No data yet
        try {
            limitoutfile = mLimitPolicyFile.startWrite();

            XmlSerializer out = new FastXmlSerializer();
            out.setOutput(limitoutfile, StandardCharsets.UTF_8.name());

            out.startDocument(null, true);
            out.startTag(null, TAG_SMS_LIMIT_POLICY_BODY);
            out.startTag(null, TAG_SMSLIMIT);
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
            LocalDateTime now = LocalDateTime.now();
            out.attribute(null, ATTR_DATE_INFO, dtf.format(now));
            Integer i = new Integer(latestcount);
            out.attribute(null, ATTR_SMSLIMIT_SMS_COUNT, i.toString());
            out.endTag(null, TAG_SMSLIMIT);
            out.endTag(null, TAG_SMS_LIMIT_POLICY_BODY);
            out.endDocument();
            mLimitPolicyFile.finishWrite(limitoutfile);
        } catch (IOException ex) {
            Rlog.e(TAG, "Unable to write premium SMS policy database", ex);
            if (limitoutfile != null) {
                mLimitPolicyFile.failWrite(limitoutfile);
            }
        }


    }

    private void updatedateandcount() {

        File dir = new File(SMS_POLICY_FILE_DIRECTORY);
        mLimitPolicyFile = new AtomicFile(new File(dir, SMS_MAXLIMIT_POLICY_FILE_NAME));
        FileOutputStream limitoutfile = null;

        // No data yet
        try {
            limitoutfile = mLimitPolicyFile.startWrite();

            XmlSerializer out = new FastXmlSerializer();
            out.setOutput(limitoutfile, StandardCharsets.UTF_8.name());

            out.startDocument(null, true);
            out.startTag(null, TAG_SMS_LIMIT_POLICY_BODY);
            out.startTag(null, TAG_SMSLIMIT);
            DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
            LocalDateTime now = LocalDateTime.now();
            out.attribute(null, ATTR_DATE_INFO, dtf.format(now));
            out.attribute(null, ATTR_SMSLIMIT_SMS_COUNT, "10");
            out.endTag(null, TAG_SMSLIMIT);
            out.endTag(null, TAG_SMS_LIMIT_POLICY_BODY);
            out.endDocument();

            mLimitPolicyFile.finishWrite(limitoutfile);
        } catch (IOException ex) {
            Rlog.e(TAG, "Unable to write premium SMS policy database", ex);
            if (limitoutfile != null) {
                mLimitPolicyFile.failWrite(limitoutfile);
            }
        }


    }


    private boolean isdatechanged() {
        File dir = new File(SMS_POLICY_FILE_DIRECTORY);
        mLimitPolicyFile = new AtomicFile(new File(dir, SMS_MAXLIMIT_POLICY_FILE_NAME));
        FileInputStream limitinfile = null;
        String count = "0";
        try {
            limitinfile = mLimitPolicyFile.openRead();
            final XmlPullParser parser = Xml.newPullParser();
            parser.setInput(limitinfile, StandardCharsets.UTF_8.name());

            XmlUtils.beginDocument(parser, TAG_SMS_LIMIT_POLICY_BODY);

            XmlUtils.nextElement(parser);
            String element = parser.getName();
            if (element.equals(TAG_SMSLIMIT)) {
                String filedate = parser.getAttributeValue(null, ATTR_DATE_INFO);
                DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy/MM/dd");
                LocalDateTime now = LocalDateTime.now();
                return !filedate.equals(dtf.format(now));
            }
        } catch (FileNotFoundException e) {

        } catch (IOException e) {
            Rlog.e(TAG, "Unable to read premium SMS policy database", e);
        } catch (NumberFormatException e) {
            Rlog.e(TAG, "Unable to parse premium SMS policy database", e);
        } catch (XmlPullParserException e) {
            Rlog.e(TAG, "Unable to parse premium SMS policy database", e);
        } finally {
            if (limitinfile != null) {
                try {
                    limitinfile.close();
                } catch (IOException ignored) {}
            }
        }
        return false;
    }
```

Identify the below mentioned function

```
    private boolean isUnderLimit(ArrayList<Long> sent, int smsWaiting) {
        Long ct = System.currentTimeMillis();
        long beginCheckPeriod = ct - mCheckPeriod;

        if (VDBG) log("SMS send size=" + sent.size() + " time=" + ct);

        while (!sent.isEmpty() && sent.get(0) < beginCheckPeriod) {
            sent.remove(0);
        }
        if ((sent.size() + smsWaiting) <= mMaxAllowed) {
            for (int i = 0; i < smsWaiting; i++ ) {
                sent.add(ct);
            }
            return true;
        }
        return false;
    }
```

replace above funtion with

```
    private boolean isUnderLimit(ArrayList < Long > sent, int smsWaiting) {

        int remainingcount;
        Long ct = System.currentTimeMillis();
        long beginCheckPeriod = ct - mCheckPeriod;
        if (VDBG) log("SMS send size=" + sent.size() + " time=" + ct);

        while (!sent.isEmpty() && sent.get(0) < beginCheckPeriod) {
            sent.remove(0);
        }
        remainingcount = remainingsmscount();
        if ((remainingcount == 10) || isdatechanged()) {
            updatedateandcount();
            remainingcount = remainingsmscount();
        }
        if (remainingcount != 0) {
            for (int i = 0; i < smsWaiting; i++) {
                sent.add(ct);
            }
            updatecount(remainingcount - 1);
            return true;
        }
        return false;
    }
```

Identify the following file `AOSP_Root/frameworks/opt/telephony/src/java/com/android/internal/telephony/SMSDispatcher.java` Identify the below mentioned line in above mentioned file path

```
.setPositiveButton(r.getString(R.string.sms_control_yes), listener)
```

delete above line in source code

Identify the following file `AOSP_Root/frameworks/base/core/res/res/values/strings.xml` Identify the below mentioned lines in above mentioned file path

```
<string name="sms_control_message">&lt;b><xliff:g id="app_name">%1$s</xliff:g>&lt;/b> is sending a large number of SMS messages. Do you want to allow this app to continue sending
    <!-- See SMS_DIALOG.  This is a button choice to allow sending the SMSes. [CHAR LIMIT=30] -->
    <string name="sms_control_yes">Allow</string>
    <!-- See SMS_DIALOG.  This is a button choice to disallow sending the SMSes. [CHAR LIMIT=30] -->
    <string name="sms_control_no">Deny</string>
```

customize above lines to below lines

```
    <string name="sms_control_message">&lt;b><xliff:g id="app_name">%1$s</xliff:g>&lt;/b> app has reached its maximum usage limit</string>
    <!-- See SMS_DIALOG.  This is a button choice to allow sending the SMSes. [CHAR LIMIT=30] -->
    <string name="sms_control_yes">Allow</string>
    <!-- See SMS_DIALOG.  This is a button choice to disallow sending the SMSes. [CHAR LIMIT=30] -->
    <string name="sms_control_no">Ok</string>
```

## 13.3 Restrict Date and Time change modification

Identify the following file `AOSP_Root/packages/apps/Settings/src/com/android/settings/datetime/AutoTimePreferenceController.java` Identify the below mentioned line in above mentioned file path

```
        Settings.Global.putInt(mContext.getContentResolver(), Settings.Global.AUTO_TIME,
                autoEnabled ? 1 : 0);
```

customize above lines to below line

```
        Settings.Global.putInt(mContext.getContentResolver(), Settings.Global.AUTO_TIME,
                autoEnabled ? 1 : 1);
```

Identify the following file `AOSP_Root/packages/apps/Settings/src/com/android/settings/datetime/AutoTimeZonePreferenceController.java` Identify the below mentioned line in above mentioned file path

```
Settings.Global.putInt(mContext.getContentResolver(), Settings.Global.AUTO_TIME_ZONE,
                autoZoneEnabled ? 1 : 0);
```

customize above lines to below line

```
Settings.Global.putInt(mContext.getContentResolver(), Settings.Global.AUTO_TIME_ZONE,
                autoZoneEnabled ? 1 : 1);
```

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with SMS control mechanism integrated.

# Chapter 14

# Android Verified Boot

## 14.1 Generation of customized keys for Android Verified Boot

In Android framework by default Android Verified Boot enabled. By default, the algorithm **SHA256_RSA4096** is used with a test key from the **AOSP_-Root/external/avb/test/data** directory

Identity the file `AOSP_Root/external/avb/test/data/testkey_rsa4096.pem`
Delete above file
Execute below command
**openssl genrsa -out testkey_rsa4096.pem 4096**
Ensure the above command result testkey_rsa4096.pem got successfully copied to directory **AOSP_Root/external/avb/test/data**

The above mentioned changes ensure that Android Verified Boot uses your own custom key in the build process.

Chapter 15

# Integrity Measurement Architecture

## 15.1 Objective

The goals of the kernel integrity subsystem are to detect if files have been accidentally or maliciously altered, both remotely and locally, appraise a file's measurement against a "good" value stored as an extended attribute, and enforce local file integrity.

## 15.2 Steps to enable IMA in android

In order to enable IMA, customization of Android kerenel is required . Identify the following file `/MSM_Root/arch/arm64/configs/wahoo_defconfig` Add below mentioned new lines at the end of the file and save it.

```
CONFIG_CMDLINE="ima_audit=1 ima_tcb ima_appraise_tcb ima_appraise=enforce"
CONFIG_CMDLINE_EXTEND=y
```

Identify the following file `/MSM_Root/security/integrity/ima/ima_policy.c`
Identify the below mentioned line

```
{.action = APPRAISE, .fowner = GLOBAL_ROOT_UID, .flags = IMA_FOWNER},
```

replace above line with below line

```
{.action = APPRAISE,  .func = MMAP_CHECK, .mask = MAY_EXEC,
 .flags = IMA_FUNC | IMA_MASK},
{.action = APPRAISE,  .func = BPRM_CHECK, .mask = MAY_EXEC,
 .flags = IMA_FUNC | IMA_MASK},
{.action = APPRAISE, .fowner = GLOBAL_ROOT_UID, .flags = IMA_FOWNER | IMA_INMASK, .mask = MAY_EXEC},
```

Execute below commands

```
$ Cd AOSP_Root
$ . build /envsetup.sh
$ lunch aosp_taimen-user
$ CD /MSM_Root/
$ make wahoo_defconfig
$ make menuconfig
```

Execute below steps
1) Click on Security option
2) Enable Integrity auditing support[ By pressing Letter Y in key board ]
3) Enable Integrity Measurement architecture [ By pressing Letter Y in key board ]
4) Enable Appraise Integrity Measurement architecture [ By pressing Letter Y in key board ]
5) Save the configuration changes in .config file by selecting Save option

Execute below commands

```
$ make -jN
```

After successful kernel build process completion, the required output file will be generated in below mentioned path with the file name Image.lz4-dtb
msm/arch/arm64/boot
copy or replace our custom build kernel output file Image.lz4-dtb into AOSP prebuild kernel path (i.e AOSP_ROOT/device/google/wahoo-kernel)

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with customized kernel.
At the end of android build process you will find some statements with tag **- - salt** . Below statement depicted for example purpose only
**–salt 7308fd608075841900024f696a8550443fde4d47a30241e4083179b295bcb1a82**
. Please save the salt value which you traced out in your build process for feature use.

## 15.3 Steps to convert sparse image to raw image file

Locate the system.img and vendor.img which are available in AOSP_Root/out/target/product/taimen
Execute below commands

```
$ cd AOSP_Root
$ . build /envsetup.sh
$ lunch aosp_taimen-user
$ simg2img out/target/product/taimen/system.img systemraw.img
$ simg2img out/target/product/taimen/vendor.img vendorraw.img
```

## 15.4 Steps to install evmctl utility in ubuntu 16.04

Copy the systemraw.img and vendorraw.img from Android build machine ( ubuntu 14.04) to our ubuntu 16.04 machine Execute below commands

```
$ git clone https://git.code.sf.net/p/linux-ima/ima-evm-utils linux-ima-ima-evm-utils
$ sudo apt-get update
$ sudo apt-get install autoconf
$ sudo apt-get install libkeyutils-dev
$ sudo apt-get install libtool
$ sudo apt-get install libssl-dev
$ sudo apt install asciidoc
$ cd linux-ima-ima-evm-utils
$ ./autogen.sh
$ ./configure
$ make
$ sudo make install
$ export LD_LIBRARY_PATH=/usr/local/lib/
$ cd
$ mkdir img
$ sudo mount -w -o loop systemraw.img /home/<user>/img
$ sudo evmctl -r ima_hash /home/<user>/img
$ sudo umount /home/<user>/img
$ sudo mount -w -o loop vendorraw.img /home/<user>/img
$ sudo evmctl -r ima_hash /home/<user>/img
$ sudo umount /home/<user>/img
```

Copy back these systemraw.img and vendorraw.img from ubuntu 16.04 machine to Android build machine ( ubuntu 14.04 )

## 15.5 Steps to convert raw image to sparse image file

Copy back these systemraw.img and vendorraw.img from ubuntu 16.04 machine to Android build machine ( ubuntu 14.04 ) to below newly created directory ( ima ) Execute below commands

```
$ cd AOSP_Root
$ . build /envsetup.sh
$ lunch aosp_taimen-user
$ mmm system/core/libsparse/
$ cd
$ mkdir ima
$ cd ima
$ img2simg systemraw.img system.img
$ img2simg vendorraw.img vendor.img
```

## 15.6 Steps to generate vbmeta.img to ensure integrity of partitions

Execute below commands

```
$ cp AOSP_Root/out/target/product/taiman/aosp_taimen-img-eng.<user>.zip .
$ unzip aosp_taimen-img-eng.<user>.zip
$ cd aosp_taimen-img-eng.<user>
$ rm system.img
$ rm vendor.img
$ rm vbmeta.img
$ cd ..
$ avbtool add_hashtree_footer --partition_size 2684354560 --partition_name system
--image system.img --salt 7308fd60807584190024f696a8550443fde4d47a30241e4083179b295b
cb1a82 --setup_as_rootfs_from_kernel
$ avbtool add_hashtree_footer --partition_size 524288000 --partition_name vendor
--image vendor.img --salt 7308fd60807584190024f696a8550443fde4d47a30241e4083179b295b
cb1a82
$ cp system.img aosp_taimen-img-eng.<user>/
$ cp vendor.img aosp_taimen-img-eng.<user>/
$ cd aosp_taimen-img-eng.<user>
$ avbtool make_vbmeta_image --output vbmeta.img --key /AOSP_Root/external/avb/test/
data/testkey_rsa4096.pem --algorithm SHA256_RSA4096 --include_descriptors_from_image
boot.img --include_descriptors_from_image system.img --include_descriptors_from_image
vendor.img --include_descriptors_from_image
dtbo.img --padding_size 4096
$ avbtool verify_image --image vbmeta.img --key /AOSP_Root/external/avb/test/data/
testkey_rsa4096.pem
```

Once integrity got successfully verified then compress all images together as flashable .zip archive which can be flashable with command **fastboot -w update flashable.zip**

# Chapter 16

# Observations during IMA implementation

**Issues faced while implementing IMA**

Implementation of IMA initially started with experimenting on user debug build. There exists three modes in IMA configuration namely fix , enforce , off

**off**- is a runtime parameter that turns off integrity appraisal verification.

**enforce** - verifies and enforces runtime file integrity.

**fix**- for non measured files, updates the 'security.ima' xattr to reflect the existing file hash.

## 16.1 Scenario 1 :

Flashing a userdebug build with ima integrated kernel in fix mode in order to measure extended attributes for files covered under the IMA policy

**Expected behaviour:** The device is expected to boot successfully and expected to set/update the missing ima extended attributes to all the files covered under the IMA policy

**Actual behaviour during execution :** It is observer that, the device booted successfully but its unable to set/update the missing IMA extended attributes.

**Reason for the abnormal behaviour:** After thorough debugging, it is observed that, since the system and vendor partitions are mounted as Read only partitions, during IMA fix mode, IMA is unable to set the extended attributes

## 16.2 Scenario 2 :

So, in order to overcome the issue as mentioned in scenario 1, the following procedure is adopted. Flashed a userdebug build with IMA integrated kernel in fix mode.Then once device is successfully booted, mounted the system and vendor partitions as read/write and set the extended attributes to all files in system and vendor partitions by opening each file at least once (IMA is calculated for each file covered under the policy in the fix mode whenever the file is opened for the first time). After successful calculation of IMA extended attributes, again turn on verity to ensure verified boot. Steps required to perform the above operations are enumerated below

**Dependencies to Execute Scenario 2 :**

1 Root user capability is required to remount the device partitions in read/write mode.

2 Verity need to be disabled to remount the partitions as read/write

3 We must CAT each and every file covered under the policy in order to calculate the hash of the same

**Expected behaviour:** The device is expected to boot successfully and expected to set/update the missing IMA extended attributes to all the files covered under the policy.

**Actual behaviour during execution :** It is observed that the device is unable to boot successfully

**Reason for the abnormal behaviour:** Since the integrity of the device is lost. Android Verified Boot (AVB) calculates the hashes of boot, system, vendor and dtbo partitions during the build process and stores the same in a file called vbmeta.img. As the extended attributes are added to the files, hashes of the partitions are changes and integrity verification failed because of which device is unable to boot

So, it is understood that there is a need for an alternative approach in order to avoid dependency on the requirement of user debug build and disabling of AVB as the final release need to be in user build and AVB enabled.

## 16.3 Scenario 3 :

Build the framework code in user build with integrate IMA enabled kernel in enforcing mode.Once build process is complete, convert the sparse images [ i.e system.img and vendor.img] to raw image format [ i.e systemraw.img and vendorraw.img]

Mount the raw partitions and then calculate IMA extended attributes using evmctl utility. Once extended attributes are successfully calculated, unmount the raw partitions and again convert them back to sparse image format. Using avbtool make vbmeta.img by taking the new sparse images which contain extended attributes and other required images as input.
You can also refer chapter 14 for the detailed process for scenario 3

**Dependencies to Execute Scenario 3 :** evmctl utility and ubuntu 16.04

**Expected behaviour:** The device is expected to boot successfully and with IMA in enforcing mode

**Actual behaviour during execution :** The device is able to boot successfully with IMA in enforcing mode.

# Chapter 17

# Ulimit Configuration

## 17.1 Configuring the ulimit values in order to avoid system freeze

**The boot-time rlimit defaults for the init task are set in the file KERNEL_ROOT/include/asmgeneric/resource.h as below**

### 17.1.1 RLIMIT_FSIZE:

The maximum size of files that the process may create. Default values of soft limit and hard limit defined in Android are as below
`[RLIMIT_FSIZE]={RLIM_INFINITY,RLIM_INFINITY}`

Ext4 has a maximum filesystem size of 1EB and maximum filesize of 16TB. FAT32 limit the size of a single file to 4 GB. So, replicating the same file size limit of 4 GB in this case also by adding following definition in the file KERNEL_ROOT/include/uapi/asmgeneric/resource.h

```
#defineFILE_SIZE_MAX(4294967296)//4GB
```

Now, replace [RLIMIT_FSIZE]values as below in the file

```
KERNEL_ROOT/include/asm-generic/resource.h
[RLIMIT_FSIZE]={FILE_SIZE_MAX,FILE_SIZE_MAX}
```
Ref:
`https://unix.stackexchange.com/questions/306913/limit-the-maximum-size-of-file-in-ext4-filesystem`
`https://www.quora.com/What-is-the-maximum-file-size-on-Windows-10`

### 17.1.2 RLIMIT_STACK:

The maximum size of the process stack, in bytes. Default values of soft limit and hard limit defined in Android are as below
`[RLIMIT_STAC]={_STK_LIM,RLIM_INFINITY}where_STK_LIMisdefinedas`
#define _STK_LIM (8*1024*1024)

Which is nothing but 8 MB. So, setting the hard limit also to 8 MB by replacing [RLIMIT_STACK] values as below in the file KERNEL_ROOT/include/asm-generic/resource.h
`[RLIMIT_STACK]={_STK_LIM,_STK_LIM}`

### 17.1.3 RLIMIT_CORE:

Maximum size of core file. The default action of certain signals is to cause a process to terminate and produce a core dump file, a disk file containing an image of the process's memory at the time of termination. When this value is 0, no core dump files are created. Default values of soft limit and hard limit defined in Android are as below.
`[RLIMIT_CORE]={0,\hspace{2pt}RLIM_INFINITY}`
So, setting the hard limit also to 0 by replacing [RLIMIT_CORE] values as below in the file
`KERNEL_ROOT/include/asm-generic/resource.h`
`[RLIMIT_CORE]={0,0}`

### 17.1.4 RLIMIT_NOFILE:

Specifies a value one greater than the maximum file descriptor number that can be opened by this process. Default values of soft limit and hard limit defined in Android are as below
`[RLIMIT_NOFILE]={INR_OPEN_CUR,INR_OPEN_MAX}`
Where
`#defineINR_OPEN_CUR1024/*Initialsettingfornfilerlimits*/`
`#defineINR_OPEN_MAX4096/*Hardlimitfornfilerlimits*/`
So, setting the hard limit also to 1024 by replacing [RLIMIT_NOFILE] values as below in the file KERNEL_ROOT/include/asm-generic/resource.h
`[RLIMIT_NOFILE]={INR\_OPEN\_CUR,INR_OPEN_CUR}`

### 17.1.5 RLIMIT_LOCKS:

Limit the no of locks for a file to some sane default. Default values of soft limit and hard limit defined in Android are as below.
`[RLIMIT_LOCKS]={RLIM_INFINITY,RLIM_INFINITY}`
9500 is the default value for windows. So, setting the default value in Android to 9500 by adding following definition in the file
KERNEL_ROOT/include/uapi/asm-generic/resource.h
`#defineFILE_LOCK_MAX(9500)`
Now, replace [RLIMIT_LOCKS] values as below in the file
KERNEL_ROOT/include/asm-generic/resource.h
`[RLIMIT_LOCKS]={FILE_LOCK_MAX,FILE_LOCK_MAX}`

### 17.1.6 RLIMIT_AS

The maximum size of the process's virtual memory (address space) in bytes. Default values of soft limit and hard limit defined in Android are as below.
`[RLIMIT_AS]={RLIM_INFINITY,RLIM_INFINITY}`

In 64-bit Windows, the theoretical amount of virtual address space is $2\hat{6}4$ bytes (16 exabytes), but only a small portion of the 16-exabyte range is actually used. The 8-terabyte range from $0x000'00000000$ through $0x$7FF'FFFFFFFF is used for user space, and portions of the 248-terabyte range from 0xFFFF0800$'$00000000 through 0xFFFFFFFF'FFFFFFFF are used for system space.
So, setting the default value in Android to 8 TB by adding following definition in the file
`KERNEL_ROOT/include/uapi/asm-generic/resource.h`
**#define AS_LIMIT_MAX (8796093022208) //8TB**
Now, replace *[RLIMIT_AS]* values as below in the file
*KERNEL_ROOT/include/asm-generic/resource.h*
`[RLIMIT_AS]={AS_LIMIT_MAX,AS_LIMIT_MAX}`
Ref:
`https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/virtual-address-spaces`

### 17.1.7 RLIMIT_DATA:

The maximum size of the process's data segment (initialized data, uninitialized data, and heap). Default values of soft limit and hard limit defined in Android are as below.
`[RLIMIT_DATA]={RLIM_INFINITY,RLIM_INFINITY}`

As, the address space size is limited to 8 TB, data and heap sizes are being limited to 4 TB. So, setting the default value in Android to 4 TB by adding following definition in the file

`KERNEL_ROOT/include/uapi/asm-generic/resource.h`
**# define DATA_SIZE_MAX (4398046511104) // 4 TB**

Now, replace *[RLIMIT_DATA]* values as below in the file

**KERNEL_ROOT/include/asm-generic/resource.h**
`[RLIMIT_DATA]={DATA_SIZE_MAX,DATA_SIZE_MAX}`

Ref:
`https://blogs.technet.microsoft.com/markrussinovich/2009/07/05/pushing-the-limits-of-windows-processes-and-threads/`

### 17.1.8 RLIMIT_NPROC:

The number of process or, more precisely on Linux, threads still existing for the real user ID of the calling process. So long as the current number of processes belonging to this process's real user ID is greater than or equal to this limit, fork(2) fails with the error EAGAIN.

Default values of soft limit and hard limit defined in Android are as below in the file `KERNEL\_ROOT/kernel/fork.c`

```
init_task.signal->rlim[RLIMIT_NPROC].rlim_cur=max_threads/2;
init_task.signal->rlim[RLIMIT_NPROC].rlim_max=max_threads/2;
```

Where the value of max_threads is 27889

So, by default, the soft limit and hard limit of *RLIMIT_NPROC* are set to 13944.
So, in order to reduce the number of threads still existing for the real user ID, change the above assignment to as below

```
init_task.signal->rlim[RLIMIT_NPROC].rlim_cur=max_threads/50;
init_task.signal->rlim[RLIMIT_NPROC].rlim_max=max_threads/50;
```

Which means, we are setting the soft limit and hard limit of RLIMIT_NPROC to 557

### 17.1.9 RLIMIT_SIGPENDING (since Linux 2.6.8):

This is a limit on the number of signals that may be queued for the real user ID of the calling process.
Default values of soft limit and hard limit defined in Android are as below in the file
`KERNEL_ROOT/msm/kernel/fork.c`
```
[init_task.signal->rlim[RLIMIT_SIGPENDING]=init_task.signal->
rlim[RLIMIT_NPROC];
```

Which means that soft limit and hard limit of RLIMIT_SIGPENDING is set to the soft limit and hard limit of RLIMIT_NPROC

As the soft limit and hard limit of RLIMIT_NPROC are already set to 557, soft limit and hard limit of RLIMIT_SIGPENDING are also set to 557

### 17.1.10 RLIMIT_MSGQUEUE (since Linux 2.6.8):

This is a limit on the number of bytes that can be allocated for POSIX message queues for the real user ID of the calling process.
Default values of soft limit and hard limit defined in Android are as below in the file
`KERNEL_ROOT/include/uapi/asm-generic/resource.h`
`[RLIMIT_MSGQUEUE]={MQ_BYTES_MAX,MQ_BYTES_MAX\}`
Where MQ_BYTES_MAX is defined as below in the file
`KERNEL_ROOT/include/uapi/linux/mqueue.h`
**#define MQ_BYTES_MAX 8192004**
In order to reduce the number of bytes that can be allocated for POSIX message queues for the real user ID of the calling process, change the assignment in the file KERNEL_ROOT/include/uapi/asm-generic/resource.h as below
`[RLIMIT_MSGQUEUE]={MQ_BYTES_MAX_NEW,MQ_BYTES_MAX_NEW\}`
And define MQ_BYTES_MAX_NEW as below in the file
`KERNEL_ROOT/include/uapi/asm-generic/resource.h`
**#define MQ_BYTES_MAX_NEW 409600**

### 17.1.11 RLIMIT_MEMLOCK:

Maximum number of bytes of memory that may be locked into RAM. This limit is in effect rounded down to the nearest multiple of the system page size. Default values of soft limit and hard limit defined in Android are as below in the file
`KERNEL_ROOT/include/uapi/asm-generic/resource.h`
`[RLIMIT_MEMLOCK]={MLOCK_LIMIT,MLOCK_LIMIT\}`
Where MLOCK_LIMIT is defined as below in the file
`KERNEL_ROOT/include/uapi/linux/resource.h`

* /

* GPG2 wants 64kB of mlocked memory, to make sure pass phrases

* and other sensitive information are never written to disk.

* /

**#define*M*LOCK_LIMIT ((PAGE_SIZE > 64*1024) ? PAGE_SIZE : 64*1024)**
As it is mentioned that GPG2 wants 64kB of mlocked memory, to make sure pass phrases and other sensitive information are never written to disk, we are leaving this limit untouched to avoid security implications

Chapter 18

# Device Encryption

Android 7.0 and later supports file-based encryption (FBE). File-based encryption allows different files to be encrypted with different keys that can be unlocked independently.

## 18.1 Direct Boot

File-based encryption enables a new feature introduced in Android 7.0 called Direct Boot. Direct Boot allows encrypted devices to boot straight to the lock screen. Previously, on encrypted devices using full-disk encryption (FDE), users needed to provide credentials before any data could be accessed, preventing the phone from performing all but the most basic of operations. For example, alarms could not operate, accessibility services were unavailable, and phones could not receive calls but were limited to only basic emergency dialer operations.
With the introduction of file-based encryption (FBE) and new APIs to make applications aware of encryption, it is possible for these apps to operate within a limited context. This can happen before users have provided their credentials while still protecting private user information.
On an FBE-enabled device, each user of the device has two storage locations available to applications:

- Credential Encrypted (CE) storage, which is the default storage location and only available after the user has unlocked the device.
- Device Encrypted (DE) storage, which is a storage location available both during Direct Boot mode and after the user has unlocked the device.

The Direct Boot API allows encryption-aware applications to access each of these areas. There are changes to the application lifecycle to accommodate the need to notify applications when a user's CE storage is unlocked in response to first entering credentials at the lock screen. Devices running Android 7.0 and above must support these new APIs and lifecycles regardless of whether or not they implement FBE. Although, without FBE, DE and CE storage will always be in the unlocked state. A complete implementation of file-based encryption on the Ext4 and F2FS file systems is provided in the Android Open Source

Project (AOSP) and needs only be enabled on devices that meet the requirements.
All the necessary packages in AOSP have been updated to be direct-boot aware. However, where device manufacturers use customized versions of these apps, they will want to ensure at a minimum there are direct-boot aware packages providing the following services:

- Telephony Services and Dialer
- Input method for entering passwords into the lock screen

## 18.2 Kernel Support

Kernel support for Ext4 and F2FS encryption is available in the Android common kernels, version 3.18 or higher.

## 18.3 Enabling file-based encryption

FBE is enabled by adding the flag
fileencryption=contents_encryption_mode[:filenames_encryption_mode] to the fstab line in the final column for the userdata partition.
contents_encryption_mode parameter defines which cryptographic algorithm is used for the encryption of file contents and filenames_encryption_mode for the encryption of filenames. contents_encryption_mode can be only aes-256-xts. filenames_encryption_mode has two possible values: aes-256-cts and aes-256-heh. If filenames_encryption_mode is not specified then aes-256-cts value is used.
Fstab files can be found in two locations in the Android Oreo source code One is ANDROID_ROOT/device/google/wahoo/fstab.hardware. Another location is ANDROID_ROOT/vendor/google_devices/taimen/vendor/etc/fstab.taimen
In both the files, search for the line
/dev/block/platform/soc/1da4000.ufshc/by-name/userdata /data ext4
errors=panic,noatime,nosuid,nodev,barrier=1,noauto_da_alloc
latemount,wait,check,formattable,
**fileencryption=ice:aes-256-heh**,eraseblk=16777216,logicalblk=4096,quota
And replace it with the line
/dev/block/platform/soc/1da4000.ufshc/by-name/userdata /data ext4
errors=panic,noatime,nosuid,nodev,barrier=1,noauto_da_alloc
latemount,wait,check,formattable, **fileencryption=aes-256-xts:aes-256-heh**,eraseblk=16777216,logicalblk=4096,quota
Android provides a reference implementation of file-based encryption, in which vold (system/vold) provides the functionality for managing storage devices and volumes on Android. The addition of FBE provides vold with several new commands to support key management for the CE and DE keys of multiple users. In addition to the core changes to use the file-based encryption capabilities in kernel, many system packages including the lockscreen and the SystemUI have been modified to support the FBE and Direct Boot features. These include:

- AOSP Dialer (packages/apps/Dialer)

- Desk Clock (packages/apps/DeskClock)
- LatinIME (packages/inputmethods/LatinIME)*
- Settings App (packages/apps/Settings)*
- SystemUI (frameworks/base/packages/SystemUI)*

## 18.4 Encryption policy

File-based encryption applies the encryption policy at the directory level. When a device's userdata partition is first created, the basic structures and policies are applied by the init scripts. These scripts will trigger the creation of the first user's (user 0's) CE and DE keys as well as define which directories are to be encrypted with these keys. When additional users and profiles are created, the necessary additional keys are generated and stored in the keystore; their credential and devices storage locations are created and the encryption policy links these keys to those directories.
The encryption policy is hardcoded into this location:
/system/extras/ext4_utils/ext4_crypt_init_extensions.cpp
It is possible to add exceptions in this file to prevent certain directories from being encrypted at all, by adding to the directories_to_exclude list. If modifications of this sort are made then the device manufacturer should include SELinux policies that only grant access to the applications that need to use the unencrypted directory. This should exclude all untrusted applications. The only known acceptable use case for this is in support of legacy OTA capabilities.

# Chapter 19

# Fixed Path Execution of Executables, Scripts, Libraries and Applications

The requirement is to allow only whitelisted executables, scripts, libraries and applications from a fixed path. In order to achieve the requirement, paths of whitelisted executables, scripts, libraries and applications are also whitelisted at the corresponding locations in the kernel source code

## 19.1 Fixed path execution of executables and scripts

In order to allow the execution of executables and scripts only from a fixed path, we need to customize the **do_execveat_common()** function implementation which is called whenever a new executable or script is executed. The function is implemented in the file KERNEL_HOME/fs/exec.c Identify the following code in the function do_execveat_common()

```
if (IS_ERR(filename))
        return PTR_ERR(filename);
```

Add the following lines of code after the above identified code

```
if(strncmp(filename->name, "/init",5) &&
        strncmp(filename->name, "/vendor/bin/",12) &&
        strncmp(filename->name, "/vendor/xbin/",13) &&
        strncmp(filename->name, "/system/bin/",12) &&
        strncmp(filename->name, "/system/sbin/",13) &&
        strncmp(filename->name, "/system/xbin/",13) &&
        strncmp(filename->name, "/sbin/",6)){
        printk("cdac: blocking execution of  %s \n", filename->name);
        retval = -EPERM;
        goto out_ret;
    }
```

Fixed path execution of libraries and applications
In order to allow the execution of applications and loading of libraries only from a fixed path, we need to implement a function do_sys_open_wrapper in the file KERNEL_HOME/fs/open.c as below and place this function just above the implementation of the function do_sys_open

```
long do_sys_open_wrapper(int dfd, const char __user *filename, int flags,
umode_t mode) {

    char *kbuf = NULL;
    size_t len;
    size_t filedatalen=-1;

    filedatalen = strlen_user(filename);
    kbuf=(char*)kmalloc(filedatalen,GFP_KERNEL);
    len = strncpy_from_user(kbuf,filename,filedatalen);

    if (!(flags & O_DIRECTORY)){

        if(string_ends_with(kbuf, ".apk")){
            if(strncmp(kbuf, "/system/app/",12) &&
            strncmp(kbuf, "/system/priv-app/",17) &&
            strncmp(kbuf, "/system/framework/",18) &&
            strncmp(kbuf, "/vendor/overlay/",16) &&
            strncmp(kbuf, "/vendor/app/",12)){
            printk("cdac: blocking execution of app %s \n", kbuf);

                return -ENOENT;
        }
       }

        if(string_ends_with(kbuf, ".so")){
            if(strncmp(kbuf, "/system/",8) &&
                strncmp(kbuf, "/dsp/",5) &&
                strcmp(kbuf, "oemconfig.so") &&
                strcmp(kbuf, "testsig-0x3f2b7334.so") &&
                strcmp(kbuf, "testsig.so") &&
                strncmp(kbuf, "/vendor/lib64/",14) ){
                printk("cdac: blocking library %s \n", kbuf);

                    return -ENOENT;
        }
       }
    }

    kfree(kbuf);
    return do_sys_open(dfd, filename, flags, mode);
}
```

Now, replace the implementation of SYSCALL_DEFINE3 and SYSCALL_DEFINE4 with the implementation as below

```
SYSCALL_DEFINE3(open, const char __user *, filename, int, flags, umode_t, mode)
{
    if (force_o_largefile())
        flags |= O_LARGEFILE;

    return do_sys_open_wrapper(AT_FDCWD, filename, flags, mode);
}

SYSCALL_DEFINE4(openat, int, dfd, const char __user *, filename, int, flags,
        umode_t, mode)
{
    if (force_o_largefile())
        flags |= O_LARGEFILE;

    return do_sys_open_wrapper(dfd, filename, flags, mode);
    }
```

This ensures that whenever a new file is opened, do_sys_open_wrapper is called. After successfully integrating the above codes, recompile the kernel and integrate the newly generated kernel image with the Android framework. Then, recompile the Android framework to generate a new framework image with fixed path execution of executables, scripts, libraries and applications enforced

Chapter 20

# Bluetooth and Wi-Fi removal at GUI

## 20.1 Identify the below file

AOSP_
ROOTframeworks/base/packages/SystemUI/src/com/android/systemui/qs/tiles/Bluetoot
hTile.java

### 20.1.1 Identify the below lines in above file

@Override
public boolean isAvailable() {
return mController.isBluetoothSupported();
}

### 20.1.2 Customize above lines as below

@Override
public boolean isAvailable()
return false;

## 20.2 Identify the below file

AOSP_ROOTframeworks/native/data/etc/handheld_core_hardware.xml

```
<feature name="android.hardware.audio.output" />
<feature name="android.hardware.camera" />
<feature name="android.hardware.location" />
<feature name="android.hardware.location.network" />
<feature name="android.hardware.sensor.compass" />
<feature name="android.hardware.sensor.accelerometer" />
<feature name="android.hardware.bluetooth" />
<feature name="android.hardware.touchscreen" />
<feature name="android.hardware.microphone" />
```

```
<feature name="android.hardware.screen.portrait" />
<feature name="android.hardware.screen.landscape" />
```

Remove "
<feature name="android.hardware.bluetooth" /> " from aboveli
After removal

```
<feature name="android.hardware.audio.output" />
<feature name="android.hardware.camera" />
<feature name="android.hardware.location" />
<feature name="android.hardware.location.network" />
<feature name="android.hardware.sensor.compass" />
<feature name="android.hardware.sensor.accelerometer" />
<feature name="android.hardware.touchscreen" />
<feature name="android.hardware.microphone" />
<feature name="android.hardware.screen.portrait" />
<feature name="android.hardware.screen.landscape" />
```

## 20.3 Identify the below file

`AOSP_ROOT`frameworks/native/data/etc/android.hardware.bluetooth.xml
**Identify the below lines in above file**

```
<permissions>
<feature name="android.hardware.bluetooth" />
</permissions>
```

Remove "
<feature name="android.hardware.bluetooth" /> " from above lines
After removal
<permissions>
<permissions>

## 20.4 Identify the below file

`AOSP_ROOT` frame-
works/base/packages/SystemUI/src/com/android/systemui/qs/tileimpl/QSFactoryImpl.java
Identify the below lines in above file

```
public QSTile createTile(String tileSpec) {
if (tileSpec.equals("wifi")) return new WifiTile(mHost);
else if (tileSpec.equals("bt")) return new BluetoothTile(mHost);
else if (tileSpec.equals("cell")) return new CellularTile(mHost);
else if (tileSpec.equals("dnd")) return new DndTile(mHost);
else if (tileSpec.equals("inversion")) return new ColorInversionTile(mHost);
else if (tileSpec.equals("airplane")) return new AirplaneModeTile(mHost);
else if (tileSpec.equals("work")) return new WorkModeTile(mHost);
else if (tileSpec.equals("rotation")) return new RotationLockTile(mHost);
else if (tileSpec.equals("flashlight")) return new FlashlightTile(mHost);
else if (tileSpec.equals("location")) return new LocationTile(mHost);
else if (tileSpec.equals("cast")) return new CastTile(mHost);
else if (tileSpec.equals("hotspot")) return new HotspotTile(mHost);
else if (tileSpec.equals("user")) return new UserTile(mHost);
else if (tileSpec.equals("battery")) return new BatterySaverTile(mHost);
else if (tileSpec.equals("saver")) return new DataSaverTile(mHost);
else if (tileSpec.equals("night")) return new NightDisplayTile(mHost);
else if (tileSpec.equals("nfc")) return new NfcTile(mHost);
// Intent tiles.
else if (tileSpec.startsWith(IntentTile.PREFIX)) return IntentTile.create(mHost, tileSpec)
else if (tileSpec.startsWith(CustomTile.PREFIX)) return CustomTile.create(mHost,
tileSpec);
else {
Log.w(TAG, "Bad tile spec: " + tileSpec);
return null;
}
}
```

Customize above lines as below

```
public QSTile createTile(String tileSpec) {
if (tileSpec.equals("cell")) return new CellularTile(mHost);
else if (tileSpec.equals("dnd")) return new DndTile(mHost);
else if (tileSpec.equals("inversion")) return new ColorInversionTile(mHost);
else if (tileSpec.equals("airplane")) return new AirplaneModeTile(mHost);
else if (tileSpec.equals("work")) return new WorkModeTile(mHost);
else if (tileSpec.equals("rotation")) return new RotationLockTile(mHost);else if (tileSpec
else if (tileSpec.equals("location")) return new LocationTile(mHost);
else if (tileSpec.equals("user")) return new UserTile(mHost);
else if (tileSpec.equals("battery")) return new BatterySaverTile(mHost);
else if (tileSpec.equals("saver")) return new DataSaverTile(mHost);
else if (tileSpec.equals("night")) return new NightDisplayTile(mHost);
else if (tileSpec.equals("nfc")) return new NfcTile(mHost);
// Intent tiles.
else if (tileSpec.startsWith(IntentTile.PREFIX)) return IntentTile.create(mHost, tileSpec)
else if (tileSpec.startsWith(CustomTile.PREFIX)) return CustomTile.create(mHost,
tileSpec);
else {
Log.w(TAG, "Bad tile spec: " + tileSpec);
return null;
}
}
```

## 20.5 Identify the below files

`AOSP_ROOT`build/target/product/generic_no_telephony.mk
`AOSP_ROOT`build/target/product/core_tiny.mk
In above files delete all lines which consists of word “Bluetooth” ( small or capitals)

## 20.6 Identify the below file

AOSP_ROOTpackages/apps/Settings/res/xml/connected_devices.xml
Delete below lines from above file

```
<com.android.settings.widget.MasterSwitchPreference
android:key="toggle_bluetooth"
android:title="@string/bluetooth_settings_title"
android:icon="@drawable/ic_settings_bluetooth"
android:order="-7"/>
```

## 20.7 Identify the below file

AOSP_ROOT packages/apps/Settings/src/com/android/settings/connecteddevice/ConnectedDeviceDashboardFragment.java
Delete below lines from above file
final BluetoothMasterSwitchPreferenceController
bluetoothPreferenceController = new
BluetoothMasterSwitchPreferenceController( context,
Utils.getLocalBtManager(context), this, (SettingsActivity) getActivity());
lifecycle.addObserver(bluetoothPreferenceController);
controllers.add(bluetoothPreferenceController);
keys.add(BluetoothMasterSwitchPreferenceController.KEY_TOGGLE_BLUETOOTH);

## 20.8 Factory reset option removal

### 20.8.1 Identify the below file

AOSP_ROOTpackages/apps/Settings/res/xml/system_dahboard_fragment.xml
Delete below lines form above file

```
 <Preference
        android:key="reset_dashboard"
        android:title="@string/reset_dashboard_title"
        android:summary="@string/reset_dashboard_summary"
        android:icon="@drawable/ic_restore"
        android:order="-20"
        android:fragment="com.android.settings.system.ResetDashboardFragment" />
```

## 20.9 Identify the below file

AOSP_ROOTpackages/apps/Settings/res/layout/settings_main_dashboard.xml
Before modification

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:orientation="vertical">
<FrameLayout
android:id="@+id/search_bar_container"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@color/suggestion_condition_background">
<android.support.v7.widget.CardView
android:id="@+id/search_bar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_margin="@dimen/search_bar_margin"
app:cardCornerRadius="2dp"
app:cardBackgroundColor="?android:attr/colorBackground"app:cardElevation="2dp">
<Toolbar
android:id="@+id/search_action_bar"
android:layout_width="match_parent"
android:layout_height="@dimen/search_bar_height"
android:background="?android:attr/selectableItemBackground"
android:contentInsetStartWithNavigation="64dp"
android:navigationIcon="@drawable/ic_search_24dp"
android:navigationContentDescription="@string/search_menu"
android:title="@string/search_menu"
android:titleTextAppearance="@style/TextAppearance.SearchBar"
android:theme="?android:attr/actionBarTheme"/>
</android.support.v7.widget.CardView>
</FrameLayout>
<FrameLayout
android:id="@+id/main_content"
android:layout_height="match_parent"
android:layout_width="match_parent"/>
</LinearLayout>
```

After modification

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">
<FrameLayout
android:id="@+id/search_bar_container"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:visibility="gone"
android:background="@color/suggestion_condition_background">
<android.support.v7.widget.CardView
android:id="@+id/search_bar"
android:layout_width="match_parent"
android:layout_height="wrap_content"
```

```
android:layout_margin="@dimen/search_bar_margin"
app:cardCornerRadius="0dp"
android:visibility="gone"
app:cardBackgroundColor="?android:attr/colorBackground"app:cardElevation="0dp">
<Toolbar
android:id="@+id/search_action_bar"
android:layout_width="match_parent"
android:layout_height="@dimen/search_bar_height"
android:background="?android:attr/selectableItemBackground"
android:contentInsetStartWithNavigation="64dp"
android:navigationIcon="@drawable/ic_search_24dp"
android:navigationContentDescription="@string/search_menu"
android:title="@string/search_menu"
android:visibility="gone"
android:titleTextAppearance="@style/TextAppearance.SearchBar"
android:theme="?android:attr/actionBarTheme"/>
</android.support.v7.widget.CardView>
</FrameLayout>
<FrameLayout
android:id="@+id/main_content"
android:layout_height="match_parent"
android:layout_width="match_parent"/>
</LinearLayout>
```

## 20.10 Identify the below file

AOSP_
ROOTpackages/apps/Settings/src/com/android/settings/network/TetherPreferenceCont
roller.java Before modification

```
@Override
public void displayPreference(PreferenceScreen screen) {
super.displayPreference(screen);
mPreference = screen.findPreference(KEY_TETHER_SETTINGS);
if (mPreference != null && !mAdminDisallowedTetherConfig) {
mPreference.setTitle(
com.android.settingslib.Utils.getTetheringLabel(mConnectivityManager));
// Grey out if provisioning is not available.
mPreference.setEnabled(!TetherSettings.isProvisioningNeededButUnavailable(mContext));
}
}
```

After modification

```
@Override
public void displayPreference(PreferenceScreen screen) {
super.displayPreference(screen);
mPreference = screen.findPreference(KEY_TETHER_SETTINGS);
if (mPreference != null && !mAdminDisallowedTetherConfig) {
mPreference.setTitle(
com.android.settingslib.Utils.getTetheringLabel(mConnectivityManager));
// Grey out if provisioning is not available.
mPreference.setEnabled(!TetherSettings.isProvisioningNeededButUnavailable(mContext));
screen.removePreference(mPreference);
}
}
```

## 20.11 Identify the below file

AOSP_ROOTpackages/apps/Settings/res/values/strings.xml
Before modification
stringname="connected_devices_dashboard_summary">Bluetooth,Cast, NFC/string¿
stringname="network_dashboard_summary_hotspot">hotspot/string¿
stringname="wifi_settings_title">Wi\u2011Fi/string¿ After modification
stringname="connected_devices_dashboard_summary">Cast,NFC/string¿
stringname="network_dashboard_summary_hotspot">/string¿
stringname="wifi_settings_title">/string¿

## 20.12 Identify the below file

AOSP_ROOTpackages/apps/Settings/res/xml/network_and_internet.xml
Identify below mentioned lines

```
<com.android.settings.widget.MasterSwitchPreference
android:fragment="com.android.settings.wifi.WifiSettings"
android:key="toggle_wifi"
android:title="@string/wifi_settings"
android:summary="@string/summary_placeholder"
android:icon="@drawable/ic_settings_wireless"
android:order="-30">
<intent
android:action="android.settings.WIFI_SETTINGS"
android:targetClass="Settings$WifiSettingsActivity"/>
</com.android.settings.widget.MasterSwitchPreference>
```

Delete above lines

## 20.13 Identify the below file

AOSP_
ROOTpackages/apps/Settings/src/com/android/settings/network/NetworkDashboardFr
agment.java
Identify below mentioned lines
final WifiMasterSwitchPreferenceController wifiPreferenceController = new
WifiMasterSwitchPreferenceController(context, metricsFeatureProvider);
lifecycle.addObserver(wifiPreferenceController);
controllers.add(wifiPreferenceController);

keys.add(WifiMasterSwitchPreferenceController.KEY_TOGGLE_WIFI);
Delete above lines

# Chapter 21

# Bootloader Protection

## 21.1 Fix of Bootloader unlock issues

Locking of Bootloader can be achieved successfully after performing below customizations. The below mentioned changes will resolve the bootloader locking and unlocking issues.
Identity the file `AOSP_Root/frameworks/base/services/core/java/com/android/server/oemlock/VendorLock.java` Identify the below mentioned line in above mentioned file path

```
return allowedByCarrier[0];
```

replace above line with below mentioned line

```
return true;
```

Identity the file `AOSP_Root/frameworks/base/core/java/android/service/oemlock/OemLockManager.java` Identify the below mentioned line in above mentioned file path

```
return mService.isOemUnlockAllowedByCarrier();
```

replace above line with below mentioned line

```
mService.isOemUnlockAllowedByCarrier();
return true;
```

Identity the file `AOSP_Root/frameworks/base/services/core/java/com/android/server/PersistentDataBlockService.java` Identify the below mentioned line in above mentioned file path

```
            if (ActivityManager.isUserAMonkey()) {
                return;
            }

            enforceOemUnlockWritePermission();
            enforceIsAdmin();

            if (enabled) {
```

```
            // Do not allow oem unlock to be enabled if it's disallowed by a user rest
            enforceUserRestriction(UserManager.DISALLOW_OEM_UNLOCK);
            enforceUserRestriction(UserManager.DISALLOW_FACTORY_RESET);
        }
```

comment above lines, after comment it will be as follows

```
      /*  if (ActivityManager.isUserAMonkey()) {
            return;
        }

        enforceOemUnlockWritePermission();
        enforceIsAdmin();

        if (enabled) {
            // Do not allow oem unlock to be enabled if it's disallowed by a user rest
            enforceUserRestriction(UserManager.DISALLOW_OEM_UNLOCK);
            enforceUserRestriction(UserManager.DISALLOW_FACTORY_RESET);
        } */
```

## 21.2 Replacing default android verified boot key

Locking of Bootloader can be achieved successfully after performing below customizations. The below mentioned changes will resolve the bootloader locking and unlocking issues.
Identity the file `AOSP_Root/frameworks/base/services/core/java/com/android/server/oemlock/VendorLock.java` Identify the below mentioned line in above mentioned file path

```
return allowedByCarrier[0];
```

replace above line with below mentioned line

```
return true;
```

Identity the file `AOSP_Root/frameworks/base/core/java/android/service/oemlock/OemLockManager.java` Identify the below mentioned line in above mentioned file path

```
return mService.isOemUnlockAllowedByCarrier();
```

replace above line with below mentioned line

```
mService.isOemUnlockAllowedByCarrier();
return true;
```

Identity the file `AOSP_Root/frameworks/base/services/core/java/com/android/server/PersistentDataBlockService.java` Identify the below mentioned line in above mentioned file path

```
        if (ActivityManager.isUserAMonkey()) {
            return;
        }

        enforceOemUnlockWritePermission();
        enforceIsAdmin();

        if (enabled) {
            // Do not allow oem unlock to be enabled if it's disallowed by a user rest
            enforceUserRestriction(UserManager.DISALLOW_OEM_UNLOCK);
            enforceUserRestriction(UserManager.DISALLOW_FACTORY_RESET);
        }
```

comment above lines, after comment it will be as follows

```
      /*  if (ActivityManager.isUserAMonkey()) {
            return;
        }

        enforceOemUnlockWritePermission();
        enforceIsAdmin();

        if (enabled) {
            // Do not allow oem unlock to be enabled if it's disallowed by a user rest
            enforceUserRestriction(UserManager.DISALLOW_OEM_UNLOCK);
            enforceUserRestriction(UserManager.DISALLOW_FACTORY_RESET);
        } */
```

Perform the subsection 1.3.5 in Chapter 1 Android source code build instruction to achieve new android operating system with Bootloader unlock issues get fixed.

## 21.3 Locking of Bootloader

Successfully to lock the bootloader the following procedure need to be followed

- In Settings, tap About phone, then tap Build number seven (7) times.
- When you see the message "You are a developer", tap the back button.
- Tap Developer options and enable USB debugging.

Reboot into the bootloader and use fastboot to lock it.

- Connect your device to your computer over USB.
- Start the device in fastboot mode with one of the following methods:

  [•] Using the adb tool: With the device powered on, execute

```
$ adb reboot bootloader
```

  [•] Using a key combo: Turn the device off, then turn it on and immediately Press and hold Volume Down, then press and hold Power.

- AVB is integration into Pixel 2 XL device

  ```
  $ fastboot erase avb_custom_key
  $ avbtool extract_public_key --key AOSP_Root/external/avb/test/data/testkey_rsa4096
  .pem --output pkmd.bin
  $ fastboot flash avb_custom_key pkmd.bin
  ```

- lock the device's bootloader by executing below command

  ```
  $ fastboot flashing lock
  ```

  The successfull execution above command results ”The steps to lock the bootloader ” on mobile device screen.
  Follow the steps on mobile device screen to complete entire process successfully.

  Once after device boot in LOCKED state (i.e Yellow screen Boot warning will come when device is booting in LOCKED State) . Execute below steps

  1) Enable the Developer options
  2) Turn off (Disable ) the OEM unlocking toogle inside Developer options
  3) Click on Settings app
  4) Click on Apps & notifications
  5) Click on App info
  6) Click on Settings
  7) Click on Storage
  8) Click on CLEAR CACHE then Click on CLEAR DATA
  Note: **Before performing this chapter ensure you have taken complete keys backup which used in kernel and framework. It is recommended to perform this chapter after all modules (Chapters) executed successfully**

# Chapter 22

# Android OS Update

## 22.1 Identify the below two files

`AOSP_ROOT`system/sepolicy/private/app.te

`AOSP_ROOT`system/sepolicy/prebuilts/api/26.0/private/app.te

In above files add below new lines to get access of sdcard

```
allow platform_app fuse:dir create_dir_perms;
allow platform_app fuse:file create_file_perms;
allow platform_app sdcardfs:dir create_dir_perms;
allow platform_app sdcardfs:file create_file_perms;

allow platform_app vfat:dir r_dir_perms;
allow platform_app vfat:file rw_file_perms;
```

## 22.2 Identify the below two files

`AOSP_ROOT`system/sepolicy/private/platform_app.te

`AOSP_ROOT`system/sepolicy/prebuilts/api/26.0/private/platform_app.te

In above two files add below new lines to get internet and os update functionality to platform_app
net_domain(platform_app)

```
#cdac
allow platform_app update_engine_service:service_manager find;
allow platform_app update_engine:binder call;
allow platform_app update_engine:binder transfer;
#cdac Write to /data/ota_package for OTA packages.
allow platform_app ota_package_file:dir rw_dir_perms;
allow platform_app ota_package_file:file create_file_perms;
allow platform_app config_gz:file { open read };
allow platform_app selinuxfs:file { open read }
```

## 22.3 Identify the below two files

AOSP_ROOTsystem/sepolicy/private/update_engine.te

AOSP_ROOTsystem/sepolicy/prebuilts/api/26.0/private/update_engine.te

In above two files add below new lines to get os update functionality

allow update_engine platform_app:binder call;

allow update_engine self:capability  dac_override dac_read_search
sys_rawio ;

## 22.4 Identify the below file

AOSP_Rootframeworks/base/data/etc/privapp-permissions-platform.xml

In above files add below new lines

```
<privapp-permissions package="com.example.cdac.expsocketapp">
        <permission name="android.permission.ACCESS_CACHE_FILESYSTEM"/>
        <permission name="android.permission.REBOOT"/>
    </privapp-permissions>
```

## 22.5 Commands to generate ota update

```
$ cd AOSP_ROOT
$ . build/envsetup.sh
$ lunch aosp_taimen-user
$ mkdir dist_output [ Execute this command if folder not created atleast once ]
$ make dist DIST_DIR=dist_output
```

After successfull execution of above commands and ota update file will be generated in the below mentioned path

AOSP_ROOTdist_output/aosp_taimen-ota-eng.root.zip

## 22.6 Create a python script file(gen_update_config.py) by copying below code into the file

```
#!/usr/bin/env python3
#
# Copyright (C) 2018 The Android Open Source Project
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
```

```
#      http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

"""
```

Given a OTA package file, produces update config JSON file.

```
Example:  tools/gen_update_config.py \\
            --ab_install_type=STREAMING \\
            ota-build-001.zip  \\
            my-config-001.json \\
            http://foo.bar/ota-builds/ota-build-001.zip
"""

import argparse
import json
import os.path
import sys
import zipfile


class GenUpdateConfig(object):
    """
    A class that generates update configuration file from an OTA package.

    Currently supports only A/B (seamless) OTA packages.
    TODO: add non-A/B packages support.
    """

    AB_INSTALL_TYPE_STREAMING = ’STREAMING’
    AB_INSTALL_TYPE_NON_STREAMING = ’NON_STREAMING’
    METADATA_NAME = ’META-INF/com/android/metadata’

    def __init__(self, package, url, ab_install_type):
        self.package = package
        self.url = url
        self.ab_install_type = ab_install_type
        self.streaming_required = (
            # payload.bin and payload_properties.txt must exist.
            ’payload.bin’,
            ’payload_properties.txt’,
        )
        self.streaming_optional = (
            # care_map.txt is available only if dm-verity is enabled.
            ’care_map.txt’,
            # compatibility.zip is available only if target supports Treble.
            'compatibility.zip',
        )
        self._config = None

    @property
    def config(self):
        """Returns generated config object."""
        return self._config

    def run(self):
        """Generates config."""
        streaming_metadata = None
        if self.ab_install_type == GenUpdateConfig.AB_INSTALL_TYPE_STREAMING:
            streaming_metadata = self._gen_ab_streaming_metadata()

        self._config = {
            '__': '*** Generated using tools/gen_update_config.py ***',
            'name': self.ab_install_type[0] + ' ' + os.path.basename(self.package)[:
            'url': self.url,
            'ab_config': streaming_metadata,
            'ab_install_type': self.ab_install_type,
        }

    def _gen_ab_streaming_metadata(self):
        """Builds metadata for files required for streaming update."""
        with zipfile.ZipFile(self.package, 'r') as package_zip:
            property_files = self._get_property_files(package_zip)

            metadata = {
                'property_files': property_files
            }

        return metadata

    def _get_property_files(self, zip_file):
        """Constructs the property-files list for A/B streaming metadata."""

        def compute_entry_offset_size(name):
            """Computes the zip entry offset and size."""
            info = zip_file.getinfo(name)
            offset = info.header_offset + len(info.FileHeader())
            size = info.file_size
            return {
                'filename': os.path.basename(name),
                'offset': offset,
                'size': size,
            }

        property_files = []
        for entry in self.streaming_required:
            property_files.append(compute_entry_offset_size(entry))
        for entry in self.streaming_optional:
            if entry in zip_file.namelist():
                property_files.append(compute_entry_offset_size(entry))

        # 'META-INF/com/android/metadata' is required
        property_files.append(compute_entry_offset_size(GenUpdateConfig.METADATA_NAM

        return property_files

    def write(self, out):
        """Writes config to the output file."""
        with open(out, 'w') as out_file:
            json.dump(self.config, out_file, indent=4, separators=(',', ': '), sort_


def main():  # pylint: disable=missing-docstring
    ab_install_type_choices = [
        GenUpdateConfig.AB_INSTALL_TYPE_STREAMING,
        GenUpdateConfig.AB_INSTALL_TYPE_NON_STREAMING]
    parser = argparse.ArgumentParser(description=__doc__,
                                     formatter_class=argparse.RawDescriptionHelpForm
    parser.add_argument('--ab_install_type',
                        type=str,
                        default=GenUpdateConfig.AB_INSTALL_TYPE_STREAMING,
                        choices=ab_install_type_choices,
                        help='A/B update installation type')
    parser.add_argument('package',
                        type=str,
                        help='OTA package zip file')
    parser.add_argument('out',
                        type=str,
                        help='Update configuration JSON file')
    parser.add_argument('url',
                        type=str,
                        help='OTA package download url')
    args = parser.parse_args()

    if not args.out.endswith('.json'):
        print('out must be a json file')
        sys.exit(1)

    gen = GenUpdateConfig(
        package=args.package,
        url=args.url,
        ab_install_type=args.ab_install_type)
    gen.run()
    gen.write(args.out)
    print('Config is written to ' + args.out)
```

```
if __name__ == '__main__':
    main()
```

Run below command to generate json configuration file for already generated ota update file
(AOSP_ROOTdist_output/aosp_taimen-ota-eng.root.zip)

```
/$ python gen_update_config.py aosp_taimen-ota-eng.root.zip <ota>.json
https://<mdm server ip>:9000/aosp_taimen-ota-eng.root.zip \\

Execution of above command results a <ota>.json file \\
```

open the ota.json and add below new lines after "ab_config":
"verify_payload_metadata": false,
"force_switch_slot": true,
Below lines depicts the sample ota.json format

```
{
    "__": "*** Generated using tools/gen_update_config.py ***",
    "ab_config": {
        "verify_payload_metadata": false,
        "force_switch_slot": true,
        "property_files": [
            {
                "filename": "payload.bin",
                "offset": 5985,
                "size": 568845805
            },
            {
                "filename": "payload_properties.txt",
                "offset": 568851848,
                "size": 154
            },
            {
                "filename": "care_map.txt",
                "offset": 509,
                "size": 205
            },
            {
                "filename": "compatibility.zip",
                "offset": 767,
                "size": 5171
            },
            {
                "filename": "metadata",
                "offset": 69,
                "size": 392
            }
        ]
    },
    "ab_install_type": "STREAMING",
    "name": "S OTA_8_1_4",
    "url": "http://10.244.0.187:9000/OTA_8_1_4.zip"
}
```

Chapter 23

# Server Deployment Document

## 23.1 Prerequisites:

Ubuntu 14.04 and above

Node.js

npm

mysql

Angular 7

## 23.2 Database:

Install mysql and create username “root” and password “cdac@123”

- $ sudo apt-get install mysql-client-core-5.5
- $ sudo apt-get install mysql-server

Create anurag database to mysql by using following commands

After installation of mysql then do First login into mysql account

- $ sudo mysql -u root -p
- $ create database anurag;

Add database from the path ~/Hardend_Deployment/MDM/MDMDB.sql to anurag

- mysql> use anurag;
- mysql> source ~/Hardend_Deployment/MDM/MDMDB.sql

To verify the database mysql> show tables;

- If lock_history and wipe_history not available in database then create it by using following cmd

CREATE TABLE ‘lock_history‘ ( ‘id‘ int(11) NOT NULL AUTO_INCREMENT, ‘name‘ varchar(255) DEFAULT NULL, ‘dt‘ datetime DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP, ‘status‘ varchar(250) DEFAULT NULL, KEY ‘id‘ (‘id‘) ) ENGINE=InnoDB AUTO_INCREMENT=1711 DEFAULT CHARSET=latin1; CREATE TABLE ‘wipe_history‘ ( ‘id‘ int(11) NOT NULL AUTO_INCREMENT, ‘name‘ varchar(255) DEFAULT NULL, ‘dt‘ datetime DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP, ‘status‘ varchar(250) DEFAULT NULL, KEY ‘id‘ (‘id‘) ) ENGINE=InnoDB AUTO_INCREMENT=1711 DEFAULT CHARSET=latin1;

## 23.3 Steps to deploy server:

Install Node.js and npm from the Ubuntu repository

- $ sudo apt update
- sudo apt install curl
- curl -sL https://deb.nodesource.com/setup_8.x -o nodesource_setup.sh
- $ sudo bash nodesource_setup.sh
- sudo apt-get install nodejs
- sudo apt-get install build-essential To verify the installation execute the following command:
- $ nodejs –version

To be able to download npm packages, you also need to install npm, the Node.js package manager. To do type:

- $ sudo apt install npm
- Verify the installation by typing:
- $ npm —version

To deploy server:

Execute below commands in the same folder

- $ npm install –save
- $ nodejs app.js
  - app.js which is provide in ~/Hardend_Deployment/MDM/server

Known Issues:

Error: CERT_UNTRUSTED

- Solution
- $ npm config set strict-ssl false Error:

Can't find module "mysql/MD5"

- Apply below process to remove this error
  - $ npm install mysql/MD5 —save
  - $ sudo npm cache clean -f
  - $ sudo npm install -g n

NOTE: Above procedure may be repeated for any similar module errors

- $ sudo n stable Then run $ nodejs app.js or node app.js Error: node app.js is not responding
  - Apply below process to remove this error
    - $ sudo ln -s /usr/bin/nodejs /usr/bin/node
    - $ npm install –save
    - $node app.js or nodejs app.js

Steps to run Frontend:
Install Angular 7

$ npm install -g @angular/cli Copy source code which is in the path ~/Hardend_Deployment/MDM/ahafrontend. Run below commands in the same folder as the source code

- $ npm install Change the server ip in environment.prod.ts file which is located mdm front end which is in ~/Hardend_Deployment/MDM/ahafrontend/src/environments/ by socketServerEndpoint:`'https://10.244.x.xx:8xxx'` To deploy front end run below command
- $ ng serve (this command is to run server with out HTTPS)
- $ ng serve –host <local ip> –port 4300 –ssl true –ssl-key ~/Hardend_Deployment/MDM/ahafrontend/ssl/apache.key –ssl-cert ~/Hardend_Deployment/MDM/ahafrontend/ssl/apache.crt

To Add SSL:

Install apache2

- $ sudo apt-get install apache2

Create ssl folder in apache2 folder which is present in /etc/apache2

- $ sudo mkdir ssl

Add apache.cert and apache.key from`~/Hardend`_Deployment/MDM/certs folder

Steps to run app server from VM:

- Run the below commands in the terminal
  - cd ~/MDM/server
  - node app.js
  - This would successfully run the app server on port 8000

- NOTE: In case the server is not run successfully because of any module dependencies, run the following commands
    - $ rm -rf node-modules
    - $ npm install –save
    - This would require connectivity to internet

Steps to run frontend server from VM

- Run the below commands in the terminal
    - cd /MDM/ahafrontend
    - ng serve –host <local ip of VM> –port 4300 –ssl true –ssl-key .ssl/apache.key –ssl-cert .ssl/apache.crt
    - This would successfully run the frontend server on port 4300
- NOTE: In case the server is not run successfully because of any module dependencies, run the following commands
    - $ rm -rf node-modules
    - $ npm install –save
    - This would require connectivity to internet

## 23.4 Steps to Configure Server Ip:

- Navigate to the web console
- Configure Server Ip is available under the username and password
- Click on it and enter the required fields and save it.
    - Select HTTP/HTTPs according to the requirement
    - Give proper port number and server ip address where server is running

**OSUpdate:**
To perform OSUpdate first we have to setup static server.

## 23.5 Steps to setup static server;

- Navigate to the server folder.
- Click on uploads folder
- In uploadsapks folder, staticserver.sh script is available, run the script.

## 23.6 Steps to update OS:

- Navigate to the web console and enter the username and password
- On Dashboard, select the OS update
- Select the list of the users by click on checkbox

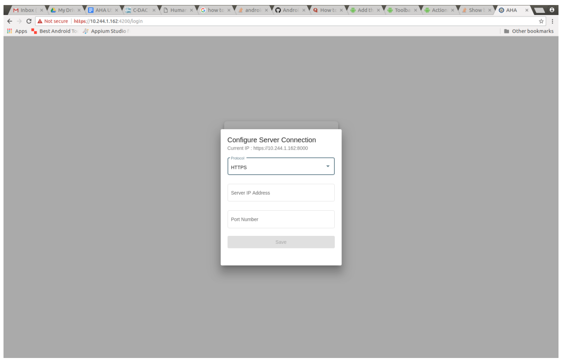
- Click on update button
    - Pre requirements:
    - User should login on mobile device
    - Check the internet connection is available or not
    - Device Administrator should be enabled

# Chapter 24

# AHA USER MANUAL

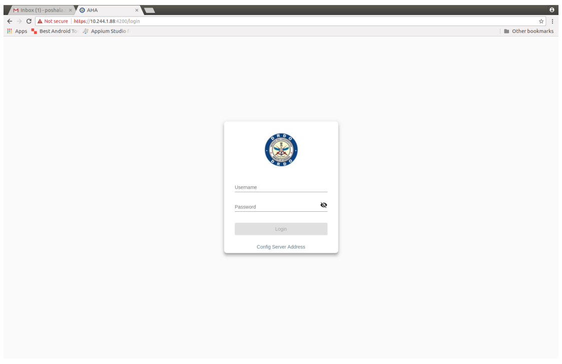
STEP 1: Configure Server Connection

- Click on Configure Server Address, and provide the details of server


Configure Server Connection
Current IP : https://10.244.1.162:8000
Protocol: HTTPS

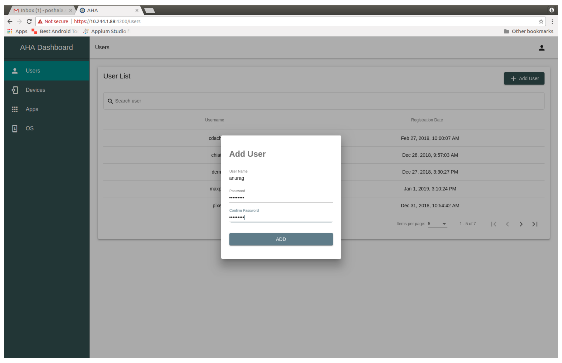
STEP 2: Login To Admin Portal

- Enter the credentials to enter into admin portal



STEP 3: Add The User in Admin Portal

- After successful login to Admin portal, click on Add user



https://10.244.1.88:4200/users

User List

Search user

| Username | Registration Date |
|---|---|
| cdach | Feb 27, 2019, 10:00:07 AM |
| chiat | Dec 28, 2018, 9:57:03 AM |
| dem | Dec 27, 2018, 3:30:27 PM |
| maxp | Jan 1, 2019, 3:10:24 PM |
| pixe | Dec 31, 2018, 10:54:42 AM |

Items per page: 5　1 - 5 of 7

Add User

User Name: anurag

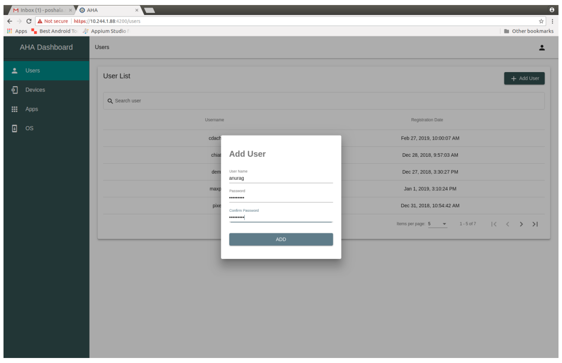
Password: ••••••••••

Confirm Password: ••••••••••

ADD

• Enter required credentials in portal

•Click on ADD


User List

Search user

| Username | Registration Date |
|---|---|
| cdach | Feb 27, 2019, 10:00:07 AM |
| chiat | Dec 28, 2018, 9:57:03 AM |
| dem | Dec 27, 2018, 3:30:27 PM |
| maxp | Jan 1, 2019, 3:10:24 PM |
| pixe | Dec 31, 2018, 10:54:42 AM |

Add User

User Name: anurag

Password: ••••••••••

Confirm Password: ••••••••••

ADD

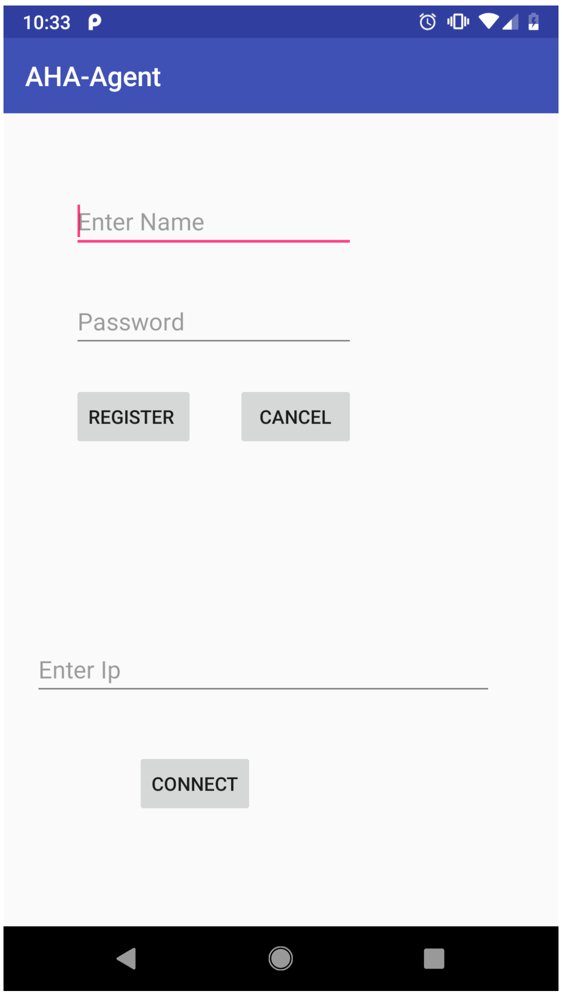
STEP 4: Registered The User from Mobile Device

•Click on AHA-Agent App

•Give the required permission to the App

NOTE: Please enable Device Admin permission presents in SETTINGS→ SECURITY–>Device Admins

•Enter server ip and click on Connect.

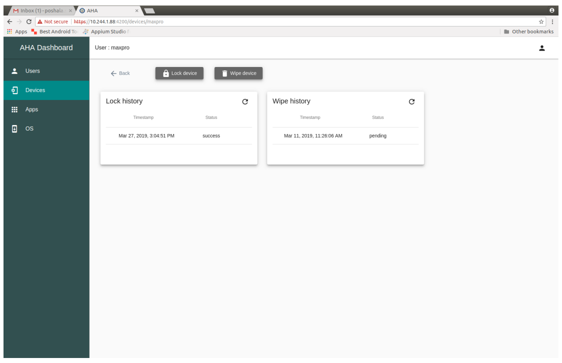


•Enter username and password which are mentioned at the time of Add user in

•Then click on Register button

STEP 5: Send Remote commands from Admin Portal

•Once connection established between client and server, then send the required remote commands from Admin portal to selected devices.



Lock history

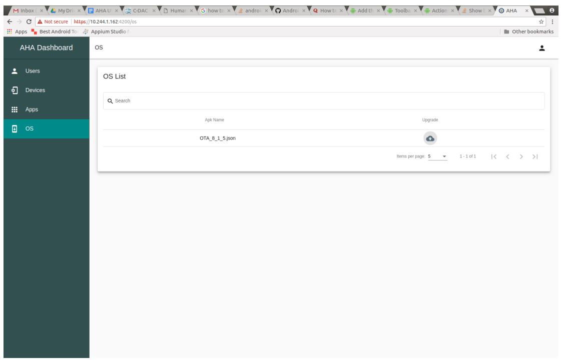
| Timestamp | Status |
|---|---|
| Mar 27, 2019, 3:04:51 PM | success |

Wipe history

| Timestamp | Status |
|---|---|
| Mar 11, 2019, 11:26:06 AM | pending |

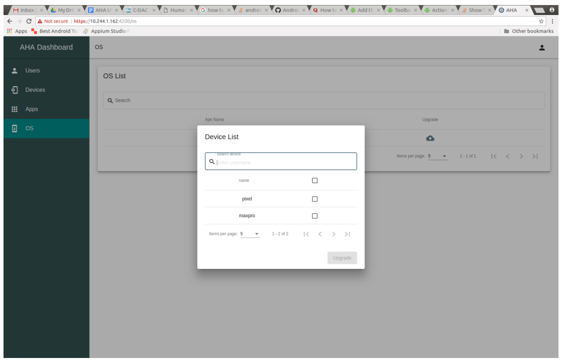
OS Update

•Select OS field from Dashboard.



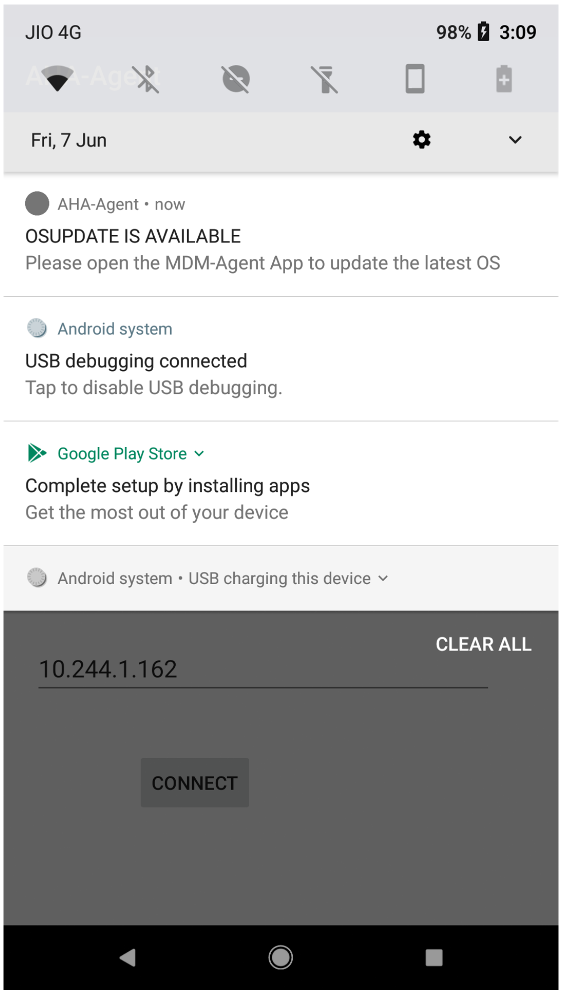
| Apk Name | Upgrade |
|---|---|
| OTA_8_1_5.json | |

•Click on the upgrade icon and select the required users to upgrade OS.



| name | ☐ |
|---|---|
| pixel | ☐ |
| maxpro | ☐ |

•Click on "OSUpdate is available" on device notification bar After OSUpdate is completed, then Reboot the device..


JIO 4G

98% 3:09

Fri, 7 Jun

AHA-Agent • now

OSUPDATE IS AVAILABLE

Please open the MDM-Agent App to update the latest OS

Android system

USB debugging connected

Tap to disable USB debugging.

Google Play Store

Complete setup by installing apps

Get the most out of your device

Android system • USB charging this device

10.244.1.162

•After OSUpdate is completed, then Reboot the device..

Chapter 25

# Integrity Measurement Architecture with OTA

## 25.1 Automatic storage of salt value in a file

In order to store the salt value automatically. Identify the following file `AOSP_Root/build/tools/releasetools/add_img_to_target_files.py` Identify the below mentioned line in above mentioned file path

```
OPTIONS.info_dict["avb_salt"] = hashlib.sha256(fp).hexdigest()
```

add below lines after above line

```
storedsalt = hashlib.sha256(fp).hexdigest()
saltpath = os.getcwd()+"/salt.txt"
file = open(saltpath, 'w')
file.write(storedsalt)
file.close()
```

## 25.2 IMA implementation for normal build

In the process of implementing IMA together with OTA we need to look for a common file structure which is available in normal build and OTA build too. During the observation it came to know `target_files_intermediates` is a critical folder need to be targeted

The above folder gets generated during normal build process

The steps involved in normal build process are

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

or

```
$ . build/envsetup.sh
$ lunch aosp_taimen-user
$ make updatepackage -jN
```

After building process successfully got completed identify the `target_files_intermediates` folder which can be located in below mentioned path

`AOSP_Root/out/target/product/taimen/obj/PACKAGING/target_files_intermediates`

Inside above folder structure we will able to find a sub folder with a name `aosp_taimen-target_files-eng.cdac` which is important as per requirement.
create a directory and create a script file inside the directory

```
$ cd
$ mkdir imacalaculation
$ cd imacalaculation
$ vi imacalaculation.sh
```

Inside script file copy the below mentioned code then save the file

```
read -p 'AOSP_Root Source code path : ' AOSP_ROOT
read -p 'Salt : ' SALT
shellscriptpath=$PWD
echo As per your input your Android Source code is IN path is  $AOSP_ROOT and the key
cd $AOSP_ROOT
. build/envsetup.sh
lunch aosp_taimen-user
cd $shellscriptpath
cd aosp*
pwd
cd IMAGES
pwd
simg2img system.img systemraw.img
simg2img vendor.img vendorraw.img
rm -rf system.img
rm -rf vendor.img
rm -rf vbmeta.img
mkdir $shellscriptpath/img
sudo mount -w -o loop systemraw.img $shellscriptpath/img
sudo evmctl -r ima_hash $shellscriptpath/img
sudo umount $shellscriptpath/img
sudo mount -w -o loop vendorraw.img $shellscriptpath/img
sudo evmctl -r ima_hash $shellscriptpath/img
sudo umount $shellscriptpath/img
img2simg systemraw.img system.img
img2simg vendorraw.img vendor.img
avbtool add_hashtree_footer --partition_size 2684354560 --partition_name system --ima
ge system.img --salt $SALT --setup_as_rootfs_from_kernel
avbtool add_hashtree_footer --partition_size 524288000 --partition_name vendor --imag
```

```
e vendor.img --salt $SALT
avbtool make_vbmeta_image --output vbmeta.img --key $AOSP_ROOT/external/avb/test/data
/testkey_rsa4096.pem --algorithm SHA256_RSA4096 --include_descriptors_from_image boot
.img --include_descriptors_from_image system.img --include_descriptors_from_image ven
dor.img --include_descriptors_from_image dtbo.img --padding_size 4096
rm -rf systemraw.img
rm -rf vendorraw.img
sudo umount $shellscriptpath/img
rm -rf $shellscriptpath/img
```

Provide execution priveleges

```
$ chmod +x imacalaculation.sh
```

Now copy the above mentioned folder i.e
`aosp_taimen-target_files-eng.cdac` into current directory
`imacalaculation` and execute below commands

```
$ cp -r AOSP_Root/out/target/product/taimen/obj/PACKAGING/target_files_intermediates
/aosp_taimen-target_files-eng.cdac .
$ sudo -s
```

Now execute the script file by providing valid inputs
`-Sourcecodepath` - `AOSP_Root`
`-Saltvalue` - which exists in `AOSP_Root/salt.txt`
After successfull execution of script file , traverse to the directory
`aosp_taimen-target_files-eng.cdac`,by executing below command

```
$ cd  aosp_taimen-target_files-eng.cdac
```

Now make a zip archive by refereing all folders present inside the folder
`aosp_taimen-target_files-eng.cdac`

The resultant zip archive will be
`aosp_taimen-target_files-eng.cdac.zip`

Now generation of final flashable zip archive with below commands by
providing above generated zip file (i.e
`aosp_taimen-target_files-eng.cdac.zip` ) as input

```
$ cd AOSP_Root
$ . build/envsetup.sh
$ lunch aosp_taimen-user
$ ./build/tools/releasetools/img_from_target_files ~/imacalaculation/aosp_taimen-tar
get_files-eng.cdac/aosp_taimen-target_files-eng.cdac.zip ~/imacalaculation/final-rele
ase.img
```

Which can be flashable into the device with below mentioned command
from fastboot mode

```
$ fastboot -w update ~/imacalaculation/final-release.img
```

## 25.3 IMA implementation for OTA build

The steps involved in OTA build process are

```
$ cd AOSP_ROOT
$ . build/envsetup.sh
$ lunch aosp_taimen-user
$ mkdir dist_output [ Execute this command if folder not created atleast once ]
$ make dist DIST_DIR=dist_output
```

After building process successfully got completed identify the `aosp_taimen-target_files-eng.cdac.zip` folder which can be identified in below mentioned path

`/AOSP_Root/dist_output/aosp_taimen-target_files-eng.cdac.zip`

uncompress the above zip file and copy the contents into `imacalaculation` folder

Before copying make sure that only script file exists inside `imacalaculation` folder,if anything found other than script file make sure to delete.

```
$ cd
$ cd imacalaculation
$ cp -r /AOSP_Root/dist_output/aosp_taimen-target_files-eng.cdac .
$ sudo -s
```

Now execute the script file by providing valid inputs
`-Sourcecodepath` - `AOSP_Root`
`-Saltvalue` - which exists in `AOSP_Root/salt.txt`
After successfull execution of script file , traverse to the directory `aosp_taimen-target_files-eng.cdac`,by executing below command

```
$ cd  aosp_taimen-target_files-eng.cdac
```

Now make a zip archive by refereing all folders present inside the folder `aosp_taimen-target_files-eng.cdac`

The resultant zip archive will be `aosp_taimen-target_files-eng.cdac.zip`

Now generation of final OTA update zip archive with below commands by providing above generated zip file (i.e `aosp_taimen-target_files-eng.cdac.zip` ) as input

```
$ cd AOSP_Root
$ . build/envsetup.sh
$ lunch aosp_taimen-user
$ ./build/tools/releasetools/ota_from_target_files ~/imacalaculation/aosp_taimen-tar
get_files-eng.cdac/aosp_taimen-target_files-eng.cdac.zip ~/imacalaculation/ota_update
```

The `ota_update.zip` is now ready for os update without loosing IMA features.