密级状态：绝密( ) 秘密( ) 内部( ) 公开(√)

Android Sepolicy配置指导

文件状态:
[ ] 草稿
[√] 正式发布
[ ] 正在修改

文件标识:	RK-KF-YF-217
当前版本:	V1.0
作者:	卞金晨
完成日期:	2019-12-06
审核:	卞金晨
审核日期:	2019-12-06

目录

1. SeAndroid 简介

1.1 SeAndroid的作用

Android的安全模型是基于一部分应用程序沙箱(sandbox)的概念, 每个应用程序都运行在自己的沙箱之中。在Android 4.3之前的版本，系统在应用程序安装时为每一个应用程序创建一个独立的uid，基于uid来控制访问进程来访问资源，这种安全模型是基于Linux传统的安全模型DAC(Discretionary Access Control，翻译为自主访问控制)来实现的。从Android 4.3开始，安全增强型Linux (SElinux) 用于进一步定义应用程序沙箱的界限。作为Android安全模型的一部分，Android使用SELinux的强制访问控制(MAC) 来管理所有的进程，即使是进程具有root(超级用户权限)的能力，SELinux通过创建自动化的安全策略(sepolicy)来限制特权进程来增强 Android的安全性。

从Android 4.4开始Android打开了SELinux的Enforcing模式，使其工作在默认的AOSP代码库定义的安全策略(sepolicy)下。在Enforcing模式下，违反SELinux安全策略的行为都会被阻止，所有不合法的访问都会记录在dmesg和logcat中。因此，我们通过查看dmesg或者logcat, 可以收集有关违背SELinux策略的错误信息，来完善我们自己的软件和SELinux策略。

1.2 SeAndroid安全策略

安全上下文实际上就是一个附加在对象上的标签（label）。这个标签实际上就是一个字符串，它由四部分内容组成，分别是SELinux用户
、SELinux 角色、类型、安全级别，每一个部分都通过一个冒号来分隔，格式
为“user:role:type:rank”。可通过ps -Z命令查看，如下所示：

```
LABEL                          USER     PID   PPID  NAME
u:r:init:s0                    root      1     0     /init
u:r:kernel:s0                  root      2     0     kthreadd
...
u:r:kernel:s0                  root      258   2     irq/322-HPH_R O
u:r:logd:s0                    logd      259   1     /system/bin/logd
u:r:healthd:s0                 root      260   1     /sbin/healthd
u:r:lmkd:s0                    root      261   1     /system/bin/lmkd
u:r:servicemanager:s0          system    262   1
/system/bin/servicemanager
u:r:vold:s0                    root      263   1     /system/bin/vold
```

最左边的那一列是进程的SContext，以第一个进程/system/bin/logwrapper的SContext为例，其值为u:r:init:s0，其中：

- u为user的意思。SEAndroid中定义了一个SELinux用户，值为u。
- r为role的意思。role是角色之意，它是SELinux中一种比较高层次，一个u可以属于多个role，不同的role具有不同的权限。
- init，代表该进程所属的Domain为init，是这个进程type，在andorid里面，定义了100多个type。按照目前我的理解，这个是进程所属的类型。
- S0是一个安全的等级MLS将系统的进程和文件进行了分级，不同级别的资源需要对应级别的进程才能访问。

1.3 SeAndroid关键文件

- 政策文件

以 *.te 结尾的文件是 SELinux 政策源代码文件，用于定义域及其标签。您可能需要在
/device/manufacturer/device-name/sepolicy 中创建新的
政策文件，但您应尽可能尝试更新现有文件。

- 上下文的描述文件

1. file_contexts 用于为文件分配标签，并且可供多种用户空间组件使用。在创建新政策时，请创建或更新该文件，以便为文件分配新标签。要应用新的 file_contexts，请重新构建文件系统映像，或对要重新添加标签的文件运行 restorecon。在升级时，对 file_contexts 所做的更改会在升级过程中自动应用于系统和用户数据分区。此外，您还可以通过以下方式使这些更改在升级过程中自动应用于其他分区：在以允许读写的方式装载相应分区后，将 restorecon_recursive 调用添加 init.board.rc 文件中。
2. genfs_contexts 用于为不支持扩展属性的文件系统（例如，proc 或 vfat）分配标签。此配置会作为内核政策的一部分进行加载，但更改可能对内核 inode 无效。要全面应用更改，您需要重新启动设备，或卸载并重新装载文件系统。此外，通过使用 context=mount 选项，您还可以为装载的特定系统文件（例如 vfat）分配特定标签。
3. property_contexts 用于为 Android 系统属性分配标签，以便控制哪些进程可以设置这些属性。在启动期间，init 进程会读取此配置。
4. service_contexts 用于为 Android Binder 服务分配标签，以便控制哪些进程可以为相应服务添加（注册）和查找（查询）Binder 引用。在启动期间，servicemanager 进程会读取此配置。
5. seapp_contexts 用于为应用进程和 /data/data 目录分配标签。在每次应用启动时，zygote 进程都会读取此配置；在启动期间，installd 会读取此配置。

6. mac_permissions.xml 用于根据应用签名和应用软件包名称（后者可选）为应用分配 seinfo 标记。随后，分配的 seinfo 标记可在seapp_contexts 文件中用作密钥，以便为带有该 seinfo 标记的所有应用分配特定标签。在启动期间，system_server 会读取此配置。

- BoardConfig.mk makefile

```
修改或添加政策文件和上下文的描述文件后，请更新您的 /device/rockchip/device-
name/BoardConfig.mk
BOARD_SEPOLICY_DIRS += \
        <root>/device/rockchip/device-name/sepolicy
(8.0+版本不需要修改以下文件)
BOARD_SEPOLICY_UNION += \
        genfs_contexts \
        file_contexts \
        sepolicy.te。
```

2. Sepolicy 相关问题确认

1. 烧入user版本的镜像后如果发现有相关功能无法正常工作了，但是烧入userdebug版本的镜像后发现可以正常工作，有可能是权限没有添加成功导致。
2. 如果出现了selinux相关的权限拒绝，则在kernel log 或者android log中都有对应的"avc: denied"，当然也可能和selinux的模式有关系，我们需要首先要确认当时SELinux 的模式, 是enforcing mode 还是 permissve mode。如果问题容易复现，我们可以先将SELinux 模式调整到Permissive mode，然后再测试确认是否与SELinux 约束相关。

```
adb shell setenforce 0
setenforce 0 设置SELinux 成为permissive模式 临时关闭selinux
setenforce 1  临时打开selinux
```

3. 另外当出现selinux相关问题，我们可以通过下面的命令来抓取对应的log：

```
adb shell logcat | grep avc
或
adb shell dmesg | grep avc
```

3. Sepolicy Rule 添加及验证

3.1 根据log推导出需要添加的权限

```
例1
audit(1444651438.800:8): avc: denied { search } for pid=158 comm="setmacaddr"
name="/" dev="nandi" ino=1 scontext=u:r:engsetmacaddr:s0
tcontext=u:object_r:vfat:s0 tclass=dir permissive=0


缺少什么权限：  缺少 search权限
谁缺少权限： engsetmacaddr
对哪个节点缺少权限： vfat
什么类型的文件： dir
最后输入的命令：
allow engsetmacaddr vfat:dir { search };

例2
```

```
auditd  (  627):  avc: denied { write } for pid=15848 comm="system_server"
name="enable" dev="sysfs" ino=9381 scontext=u:r:zygote:s0
tcontext=u:object_r:sysfs:s0 tclass=file permissive=1

缺少什么权限：  缺少 write 权限
谁缺少权限： scontext=u:r:zygote:s0
对哪个文件缺少权限： tcontext=u:object_r:sysfs:s0
什么类型的文件：  tclass=file
最后输入的命令：
allow zygote sysfs:file write

例3

audit(1441759284.810:5): avc: denied { read } for pid=1494 comm="sdcard"
name="0" dev="nandk" ino=245281 scontext=u:r:sdcardd:s0
tcontext=u:object_r:system_data_file:s0 tclass=dir permissive=0

缺少什么权限：  缺少 read 权限
谁缺少权限： sdcardd
对哪个文件缺少权限： system_data_file
什么类型的文件： dir
最后输入的命令
allow sdcardd system_data_file:dir read

例4 -- 针对ioctl的特殊说明

 (avc: denied { ioctl } for path="/dev/cmx_ddlsw" dev="tmpfs" ino=10422
ioctlcmd=4d02 scontext=u:r:system_server:s0
tcontext=u:object_r:cmx_ddlsw_device:s0 tclass=chr_file permissive=0)

在rules中添加ioctl后，还需要声明具体的iocmd，必须结合源代码添加，例如：
allow system_server cmx_ddlsw_device:file { ioctl };
allowxperm system_server cmx_ddlsw_device:file ioctl { 0x4d02 };
```

- **注意**

1. 有时候avc denied的log不是一次性显示所有问题，要等解决一个权限问题之后，才会提示另外一个权限问题，比如提示缺失某个目录的read权限，你加入read之后，再显示缺少write权限，要你一次一次试，一次一次加。所以建议在permissive模式下进行debug和添加权限，否则会无限放大工作量。
2. 要加入的权限很多时，可以用中括号或者使用宏(**强烈建议使用宏，兼容性更好**)，比如allow engsetmacaddr vfat:dir { search write add_name create};

3.2 利用audit2allow 简化方法添加log

1. 从 logcat 或串口中提取相应的 avc-denied log，下面的语句为提取所有的 avc- denied log

```
$ adb shell "cat /proc/kmsg | grep avc" > avc_log.txt
```

2. 使用 audit2allow 工具生成对应的 policy 规则
   audio2allow 使用必须先 source build/envsetup.sh，导入环境变量

```
$ audit2allow -i avc_log.txt
```

3. 将对应的policy 添加到 te 文件中
   一般添加在 /device/<company>/common/sepolicy 或者
   /device/<company>/$DEVICE/sepolicy 目录下，具体哪个目录，请执行get_build_var 查看，参考第5章

3.3 使修改生效

- 根据上述log进入$SDK/device/rockchip/common，将相应的权限语句添加到相应的文件中，重新编译烧写vendor.img(8.0及以上, 之前的版本是boot.img)进行验证。对于10.0, 请通过fastboot烧写odm.img

4. GMS相关测试

```
GtsSecurityHostTestCases
CtsSecurityHostTestCases
```

5. 添加新的开机服务的权限

5.1 添加一个系统框架服务，例redmine：179485

用于针对java的binder service，如果需要调用到框架的某些接口，或是给app提供某些特殊接口，可能需要增加java binder服务：

```
    @Override
    public void onStart() {
        publishBinderService("ccc_service", new BinderService());

        publishLocalService(CCCServiceInternal.class, new LocalService());

        Watchdog.getInstance().addMonitor(this);
        Watchdog.getInstance().addThread(mHandler);
    }
```

执行命令:'get_build_var BOARD_PLAT_PRIVATE_SEPOLICY_DIR' 后对应的目录修改，如：

修改 device/rockchip/common/sepolicy/private/service.te
在最后一行添加（关联上下文）

```
type ccc_service, system_api_service, system_server_service,
service_manager_type;
```

然后修改同目录下device/rockchip/common/sepolicy/private/service_contexts 文件（给服务打上标签，可以在service_context目录进行）
中间插入一行

```
ccc u:object_r:ccc_service:s0
```

5.2 添加一个开机自启动二进制服务 - Android 7.1

- 增加某个二进制程序或某个脚本，想要开机自动执行，如 rockchip.sample.nougat.sh

```
#! /system/bin/sh
echo "starting service..."
```

注：如果是脚本，务必保持首行有shell的路径，如上所示，且中!后有空格
本节所有要修改的文件都位于：get_build_var BOARD_SEPOLICY_DIRS
执行结果列出的文件夹中。

- 声明服务

在init.rc(任意会加载执行到的rc文件，或自己声明一个新的rc文件)中声明服务

```
service rockchip-sample-nougat-sh /system/bin/rockchip.sample.nougat.sh     #格式：service service_name service_path，注意，脚本务必要可执行，服务名后接的直接就是可执行程序，
                                                                            #这个可执行程序必须在file_context中指定标签，脚本程序不能写成 /system/bin/sh xxx.sh，这样就代表服务的主体是sh/shell，而不是你的脚本!!!
    class main                                                              #class代表启动阶段，具体需求请自行搜索修改
    user root
    group root wifi
    oneshot                                                                 #只启动一次就退出；如果不是自启动，需要触发需要加 disable 参数，通过start servic_name即可启动服务
```

- 错误脚本服务示例:

```
service rockchip-sample-nougat-sh /system/bin/sh
/system/bin/rockchip.sample.nougat.sh
    class main
    user root
    group root wifi
    oneshot
```

- 声明运行域

sepolicy目录下新建 rockchip_sample_nougat.te

```
type rockchip_sample_nougat, domain;
type rockchip_sample_nougat_exec, exec_type, file_type;

init_daemon_domain(rockchip_sample_nougat)
```

- 给二进制程序打标签，绑定到te文件

在sepolicy目录下的file_contexts文件添加:

```
/system/bin/rockchip\.sample\.nougat\.sh
u:object_r:rockchip_sample_nougat_exec:s0       # 格式：service_path
u:object_r:te_type_name_exec:s0 ，要和上面声明的te文件相对应，注意转移字符 \
```

至此，结束。

5.3 添加一个开机自启动二进制服务 - Android 8.0及以上

```
注：本节所有要修改的文件都位于：get_build_var BOARD_SEPOLICY_DIRS
执行结果列出的文件夹中。

以xxxservice为例。与Android 7.1没有很大区别，注意程序或脚本要放在vendor/bin/下：
1. init.rc或其他rc
service xxxservice /vendor/bin/xxxservice
    class main
```

```
    oneshot

2. xxxservice.te，与Android 7.1没有很大区别，注声明exec type的时候要同时关联
vendor_file_type：

type xxxservice, domain;
type xxxservice_exec, exec_type, vendor_file_type, file_type;
init_daemon_domain(xxxservice)


3. file_contexts，与Android 7.1一致
添加一条
/vendor/bin/xxxservice u:object_r:xxxservice_exec:s0
```

5.4 Android 7.1~9.0添加一个新二进制服务的完整代码示例(供内部同事参考)

```
https://10.10.10.29/#/q/topic:%22npu+monitor+service%22
```

5.5 Android 8.0以上添加一个HAL服务

- 关于HAL服务

```
Android 8.0推出了新的hwservice，具体概念请参考Google官网。一般用来增加新的硬件，如NFC/指纹/虹膜识别等模块。
这里以添加NXP公司的NFC模块为例，说明如何增加新的HAL服务
```

- 添加instance节点

```
get_build_var DEVICE_MANIFEST_FILE,
修改执行结果列出的文件，增加自己的服务节点，如： device/rockchip/common/manifest.xml

 <hal format="hidl">
        <name>vendor.nxp.nxpnfc</name>
        <transport>hwbinder</transport>
        <version>1.0</version>
        <interface>
            <name>INxpNfc</name>
            <instance>default</instance>
        </interface>
    </hal>
```

- 通过hwservice统一调用，rc文件写法

```
service vendor.nxp.nxpnfc-1-0 /vendor/bin/hw/vendor.nxp.nxpnfc@1.0-service

    class hal
    user system
    group system
    # 如果在rc文件中添加了 'class hal'，即归类为hal服务，会在init的start hal阶段通过
hwservice启动所有的hal服务。
```

- 声明为hwservice

```
get_build_var BOARD_SEPOLICY_DIRS
修改执行结果列出的文件夹中的
hwservice.te:
type vnd_nxpnfc_hwservice, hwservice_manager_type;
```

- 给service文件打标签绑定

```
get_build_var BOARD_SEPOLICY_DIRS
修改执行结果列出的文件夹中的
file_contexts:
/vendor/bin/hw/vendor\.nxp\.nxpnfc@1\.0-service u:object_r:nxpnfc_hal_exec:s0
```

- 绑定到manifest 的instance节点

```
get_build_var BOARD_SEPOLICY_DIRS
修改执行结果列出的文件夹中的
hwservice_contexts:
vendor.nxp.nxpnfc::INxpNfc （对照manifest中增加的instance，别写错）
u:object_r:vnd_nxpnfc_hwservice:s0
```

- 服务的声明

```
get_build_var BOARD_SEPOLICY_DIRS
执行结果列出的文件夹中新建
nxpnfc.te:
type nxpnfc_hal, domain;
type nxpnfc_hal_exec, exec_type, vendor_file_type, file_type;
init_daemon_domain(nxpnfc_hal)
add_hwservice(nfc, vnd_nxpnfc_hwservice)  # 如果是通过nfc进程启动新加的服务，才需要添加
```

6. Android 8.0+常见问题及处理方法

6.1 怎么在Android 8.0以上的版本关闭selinux？

```
在8.0及以后的版本无法关闭，只能以permissive模式运行，如果看到疑似selinux报错的信息，看下此log的末尾，如有permissive=0，说明此log有效。
permissive=1说明只是提示，不会造成问题。
adb shell getenforce，结果显示permissive即兼容模式，enforcing则强制模式。
```

6.12 如何修改selinux为permissive模式？

```
添加androidboot.selinux=permissive到cmdline中即可，具体怎么添加，需要确认uboot的分支：
cd u-boot; git branch -a
develop -> 添加参数到parameter.txt中
next-dev -> 添加到kernel的dts中，例如px30-android.dtsi，chosen节点的bootargs值中。
```

6.13 如何在user固件下也设置selinux为permissive模式？

```
根据6.12节中所述修改后，再到system/core下增加如下修改：
kenjc@ubuntu:~/1_RK3326_P_29/system/core/init$ git diff
diff --git a/init/Android.mk b/init/Android.mk
```

```
index c4a6a50..58161ca 100644
--- a/init/Android.mk
+++ b/init/Android.mk
@@ -14,7 +14,7 @@ init_options += \
 else
 init_options += \
     -DALLOW_LOCAL_PROP_OVERRIDE=0 \
-    -DALLOW_PERMISSIVE_SELINUX=0 \
+    -DALLOW_PERMISSIVE_SELINUX=1 \
     -DREBOOT_BOOTLOADER_ON_PANIC=0 \
     -DWORLD_WRITABLE_KMSG=0 \
     -DDUMP_ON_UMOUNT_FAILURE=0
```

6.2 为什么在Android permissive模式下，有些app仍然无法访问具体节点(添加了权限后仍然提示没权限)？

敏感权限的特征：log中带有c512,c768字样

试着在untrusted_app.te 中添加了
`allow untrusted_app audio_device:chr_file { open write read };`
还是报如下权限错误：

```
[ 141.935275] type=1400 audit(1546939304.786:43): avc: denied { write } for
pid=1836 comm="Thread-4" name="pcmC0D1c" dev="tmpfs" ino=11947
scontext=u:r:untrusted_app:s0:c512,c768 tcontext=
u:object_r:audio_device:s0 tclass=chr_file permissive=1
```

先确认需要访问的节点是否为audio_device，这个节点属于敏感权限，可以的话请修改访问的目录和文件，缩小audio_device的范围
方法为1：确定访问的节点位置，通过源码或者log确定到底访问的哪一个具体的节点， 例如 /dev/pcmc00xx
2：在相应的te文件中新声明一个节点名称，如 `file.te: type test_audio_device, dev_type;`
3：在file_context中将具体节点绑定新的节点名称，如： `file_context: /dev/pcmc00xx u:object_r:test_audio_device:s0`
4. 增加或修改需要的权限：`allow untrusted_app test_audio_device:chr_file { open write read };`
如果不过GMS认证，敏感权限(c512,c768)可以直接把对象关联mlstrustedobject，但不推荐这样修改，会造成严重的安全问题：
例如：`typeattribute audio_device mlstrustedobject;`

6.3 为什么在Android 8.0+中自己添加的服务无法启动

参照第五章，排查问题。注意8.0以上版本，新建服务要放到vendor。

6.4 添加权限后编译报错，neverallow xxx

根本原因是赋予的权限过大，缩小范围即可，解决方法与第2节一致，缩小范围(声明和指定具体需要访问的节点)后即可编译通过。

6.5 Android 8.0以后，二进制服务位于vendor分区时，代码中使用system()等函数调用系统命令无效

vendor下的服务，env PATH被复写成/vendor/bin:/vendor/xbin了，所以无法识别出system分区下的命令。

```
cc_binary {
    name: "sh_vendor",
    defaults: ["sh-defaults"],
    stem: "sh",
    vendor: true,
    cflags: [
        // Additional flags for vendor variant
        "-UMKSH_DEFAULT_PROFILEDIR",
        "-UMKSHRC_PATH",
        "-UMKSH_DEFAULT_EXECSHELL",
        "-DMKSH_DEFAULT_PROFILEDIR=\"/vendor/etc\"",
        "-DMKSHRC_PATH=\"/vendor/etc/mkshrc\"",
        "-DMKSH_DEFAULT_EXECSHELL=\"/vendor/bin/sh\"",
        "-DMKSH_DEFPATH_OVERRIDE=\"/vendor/bin:/vendor/xbin\"",
    ],
}
```

调用前可以先导入system的path，如：`system("export PATH=$PATH:/system/bin; input tap 0 0");`
注意：system函数实现是fork子进程，不能单独执行export再执行input，一定要连一起执行；

6.6 我需要修改哪个目录下的文件？修改后怎么生效？

所有需要修改的文件全部通过每一节中提到的命令get_build_var查看，在8.0及后续版本**都不需要**修改system/sepolicy目录，要修改的文件没有就自己新建！
编译结果：BOARD_SEPOLICY_DIRS/BOARD_PLAT_PUBLIC_SEPOLICY_DIR -> vendor/etc/selinux; BOARD_PLAT_PRIVATE_SEPOLICY_DIR -> system/etc/selinux;

6.7 为什么ioctl的权限加上了还一直提示denied？

ioctl权限的声明需要指出具体的cmd。因为ioctl是直接操作驱动，包含上千种命令，所以必须指出具体的cmd。
具体修改方法及说明和例子，请查看第3.1节的例4。