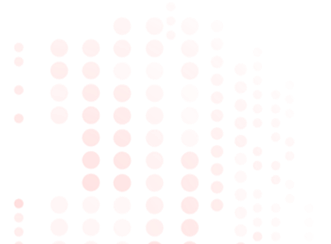


**Selinux权限修改\_V1.01**

**模组**



**芯讯通无线科技(上海)有限公司**

上海市长宁区临虹路289号3号楼芯讯通总部大楼

电话：86-21-31575100

技术支持邮箱：support@simcom.com

官网：www.simcom.com

|  |  |
| --- | --- |
| **名称：** | Selinux权限修改 |
| **版本：** | 1.01 |
| **日期：** | 2023.02.14 |
| **状态：** | 已发布 |

**版权声明**

本手册包含芯讯通无线科技（上海）有限公司（简称：芯讯通）的技术信息。除非经芯讯通书面许可，任何单位和个人不得擅自摘抄、复制本手册内容的部分或全部，并不得以任何形式传播，违反者将被追究法律责任。对技术信息涉及的专利、实用新型或者外观设计等知识产权，芯讯通保留一切权利。芯讯通有权在不通知的情况下随时更新本手册的具体内容。

本手册版权属于芯讯通，任何人未经我公司书面同意进行复制、引用或者修改本手册都将承担法律责任。

**芯讯通无线科技(上海)有限公司**

上海市长宁区临虹路289号3号楼芯讯通总部大楼

电话：86-21-31575100

邮箱：simcom@simcom.com

官网：www.simcom.com

**了解更多资料，请点击以下链接：**

http://cn.simcom.com/download/list-230-cn.html

**技术支持，请点击以下链接：**

http://cn.simcom.com/ask/index-cn.html 或发送邮件至support@simcom.com

*版权所有 © 芯讯通无线科技(上海)有限公司2021，保留一切权利。*

# 关于文档

## 版本历史

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **作者** | **备注** |
| 1.0 | 2022.5.4 | 刘明敏 | V1.00 |
| 1.1 | 2023.02.14 | 杨华坤 | V1.01 |

## 适用范围

本文档用于需要解决selinux权限问题的人员。

**目录**

[关于文档 3](#_Toc12823)

[版本历史 3](#_Toc15582)

[适用范围 3](#_Toc333)

[4](#_Toc26279)

[1 介绍 5](#_Toc15320)

[1.1 本文目的 5](#_Toc5367)

[1.2 术语和缩写 5](#_Toc1587)

[2 SELinux简介 6](#_Toc26786)

[3 SELinux 权限介绍 6](#_Toc23330)

[3.1 SELinux查看方法 6](#_Toc16812)

[3.2 SELinux工作模式 7](#_Toc29085)

[3.3 SELinux安全策略格式 7](#_Toc7698)

[3.4 SELinux权限说明 8](#_Toc28425)

[1.平台公有策略（platform public seoplicy） 8](#_Toc6036)

[2.平台私有策略（platform private seoplicy） 8](#_Toc12367)

[4 audit2allow 工具的使用方法 8](#_Toc16636)

[4.1 安装方法 8](#_Toc2190)

[4.2 参数介绍 9](#_Toc26234)

[4.3 用法 9](#_Toc2686)

[5 SELinux 常见错误分析 11](#_Toc4622)

[5.1 无SELinux权限 11](#_Toc22969)

[5.2 编译出错 11](#_Toc31768)

[5.3 修改的文件不一致 12](#_Toc11143)

[5.4 对应的te文件未找到 12](#_Toc4483)

[5.5 其它细节 13](#_Toc6888)

# 

# 介绍

## 本文目的

本文主要介绍selinux权限问题。

参考此应用文档，开发者可以很快理解并快速开发相关业务。

## 术语和缩写

|  |  |  |
| --- | --- | --- |
| 名称 | 全称 | 定义 |
|  |  |  |

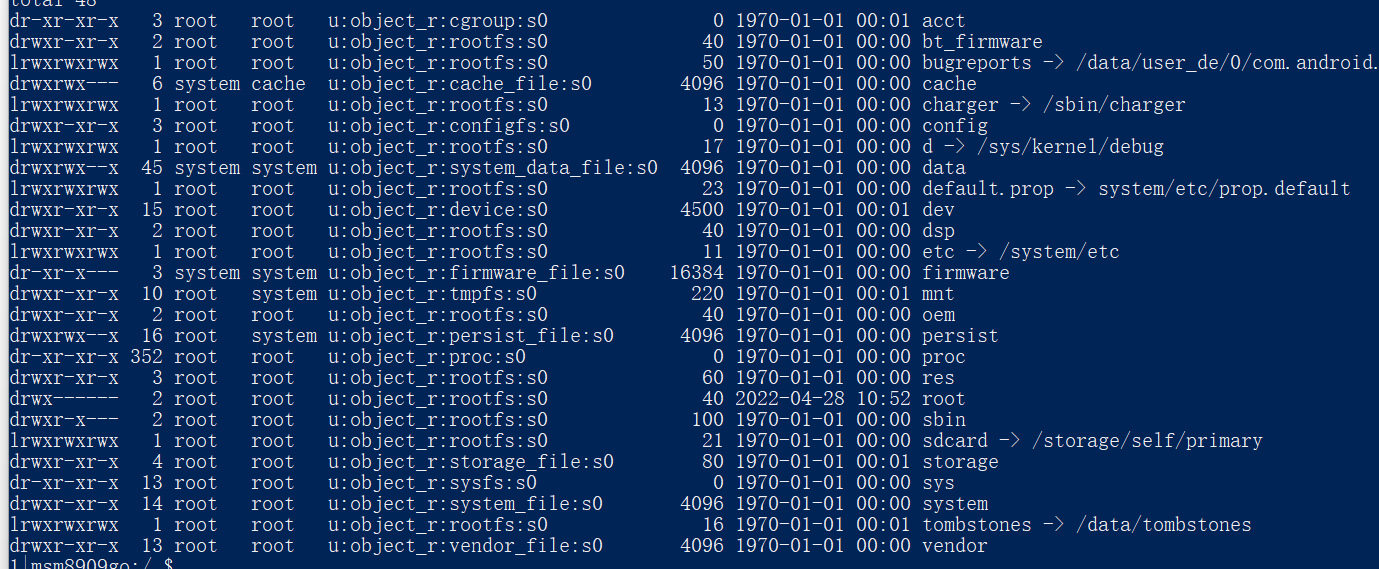
# SELinux简介

SELinux出现之前，Linux上的安全模型叫DAC，全称是Discretionary Access Control，翻译为自主访问控制。DAC的核心思想很简单，就是：进程理论上所拥有的权限与执行它的用户的权限相同。例如，以root用户启动Browser，那么Browser就有root用户的权限，在Linux系统上能做任何事情。

显然，DAC太过宽松了，所以都想方设法要在Android系统上搞到root权限。那么SELinux如何解决这个问题呢？原来，它在DAC之外，设计了一个新的安全模型，叫MAC（Mandatory Access Control），翻译为强制访问控制。MAC的处世哲学非常简单：即任何进程想在SELinux系统中做任何事情，都必须先在安全策略配置文件中赋予权限。凡是没有出现在安全策略配置文件中的权限，进程就没有该权限。

# SELinux 权限介绍

## SELinux查看方法

Linux中，每种东西都会被赋予⼀个安全属性，官⽅说法叫Security Context。Security Context（以后⽤SContext表⽰）是⼀个字符串，主要由三部分组成。例如SEAndroid中，文件或目录的SContext可通过ls -lZ命令查看，如下图所⽰：

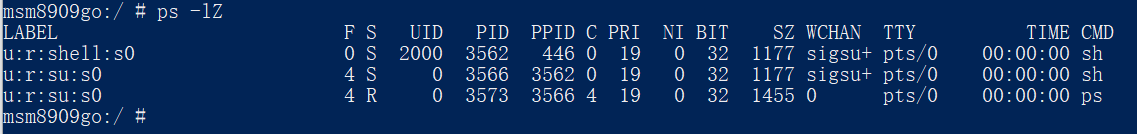
以目录acct的SContext为例，其值为u:object\_r:cgroup:s0，其中：

u为user的意思。SEAndroid中定义了⼀个SELinux⽤户，值为u。

object\_r：⽂件没法扮演⾓⾊，所以在SELinux中⽤object\_r来表⽰它的role

cgroup：和进程的Domain其实是⼀个意思。它表⽰acct⽬录对应的Type是cgroup。

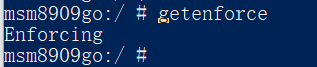
s0：MLS的级别。

进程查看权限如下：

## SELinux工作模式

Enforcing：强制模式。违反SELinux规则的行为将被阻止并记录到日志中。

Permissive：宽容模式。违反SELinux 规则的行为只会记录到日志中。一般为调试用。

查看当前工作模式：getenforce ， 命令如下:

修改当前模式：setenforce

C:\Users\fqm\AppData\Local\Temp\企业微信截图_16518257452197.png更改当前的SELINUX的模式值，后面可以跟 enforcing,permissive 或者1, 0。

Enforcing：seLinux已经打开；

Permissive：seLinux已经关闭；

setenforce 0(permissive)设置成关闭，setenforce 1(enforcing)设置成打开

## SELinux安全策略格式

SELinux中安全策略文件有自己的一套语法格式：

**rule\_name source\_type target\_type : class operation**

最常见的allow的语句，格式一般为：

**allow scontext tcontext:tclass operation**

允许scontext域的进程对tcontext类型的tclass对象去执行operation 操作，

例如：allow netd proc:file write：允许netd域中的进程对proc类型的file进行写操作，

allow：TE的allow语句，表示授权。除了allow之外，还有allowaudit、dontaudit、neverallow等。

netd：source type。也叫subject，domain。

proc：target type。它代表其后的file所对应的Type。

file：代表Object Class。它代表能够给subject操作的一类东西。例如File、Dir、socket等。在Android系统中，有一个其他Linux系统没有的Object Class，那就是Binder。

write：在该类Object Class中所定义的操作。

## SELinux权限说明

system/sepolicy/Android.mk中有对[private](https://so.csdn.net/so/search?q=private&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/mjfh095215/article/details/_blank)、public、vendor目录的说明。

在Android8.0以上，SELinux策略分离成平台（platform）和非平台（non-platform）两部分，而平台策略为了给非平台作者导出特定的类型和属性，又分为平台私有（platform private）和平台公有（platform public）部分。

**1.平台公有策略（platform public seoplicy）**

平台共有策略全部定义在/system/sepolicy/public下，public下的type和attribute可以被non-platform中的策略所使用，也就是说，设备制造商的sepolicy作者在non-platform下可以对platform public sepolicy的策略进行扩展。

**2.平台私有策略（platform private seoplicy）**

与公有策略相反，被声明为私有策略的type或attribute对non-platform的策略作者是不可见的。

举例，以8.0版本的aosp源代码中的/system/sepolicy/private/目录下的atrace.te文件为例。

1）system/sepolicy/private/file\_contexts有定义“/system/bin/atrace u:object\_r:atrace\_exec:s0”

2）system/sepolicy/private/atrace.te有定义atrace相关的规则

3）我们在device/qcom/sepolicy/common目录下新增一个atrace.te文件，并添加规则 "allow atrace sdcardfs:file read;"

当我们make进行编译时会在校验的时候失败，提示我们”device/qcom/sepolicy/common/atrace.te:2:ERROR 'unknown type atrace' at token ';' on line 23355”，那么也就是说private策略中的type和attribute对我们是不可见的。

注： 自我认知说明，scontext只有平台和非平台之分，也就是说，定义在平台private下面file\_contexts属性，public和非平台的也可以共享它。

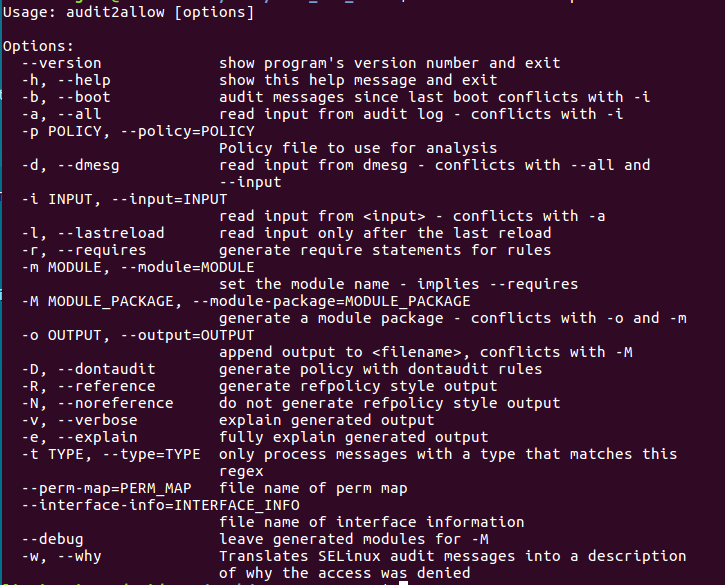
# audit2allow 工具的使用方法

audit2allow 工具可以用来生成te语句。

## 安装方法

sudo apt install policycoreutils

## 参数介绍



-p 指定一个policy文件;

-i 需要进行分析的日志文件。

## 用法

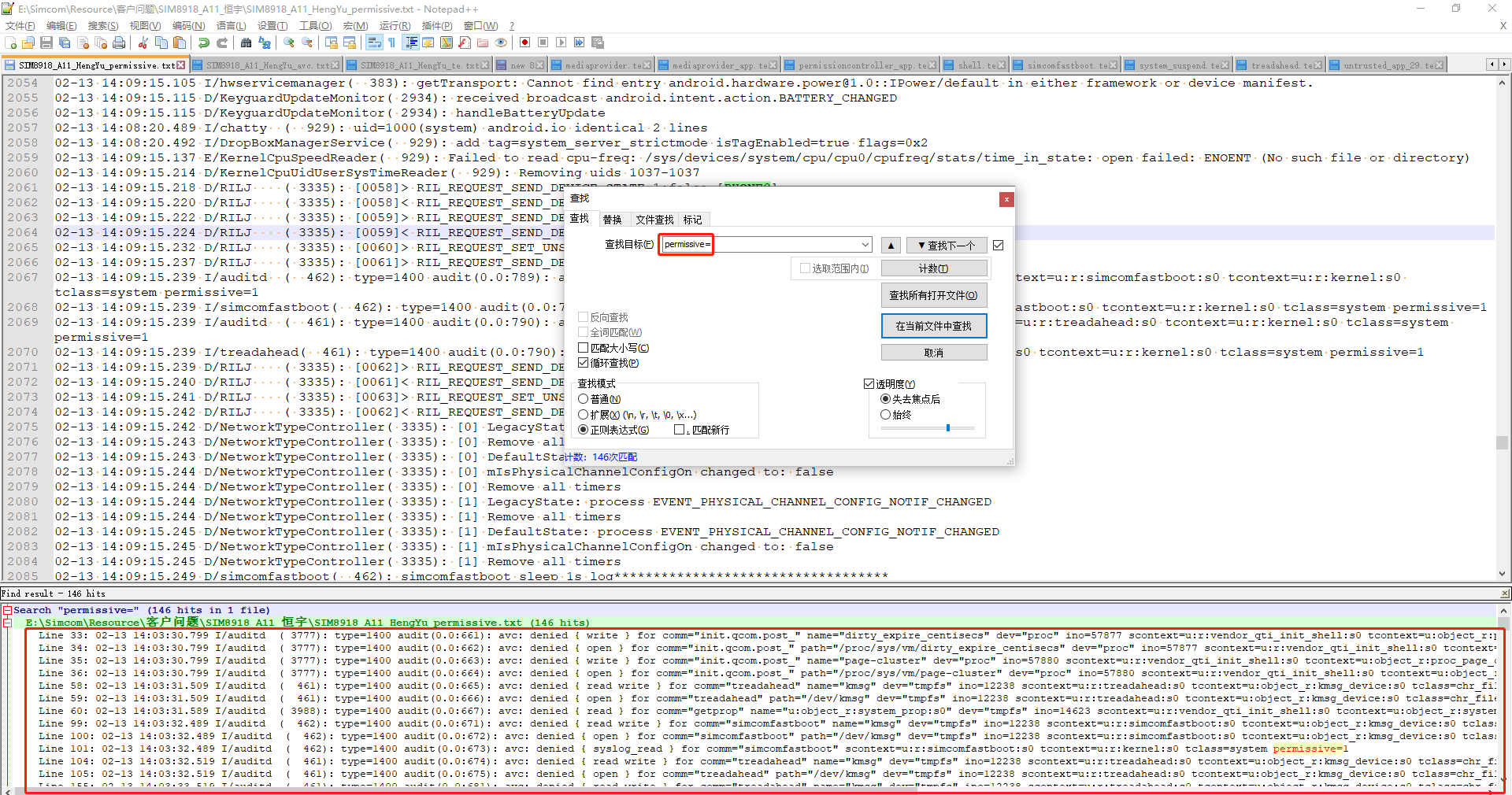
1. 抓取adb logcat log：

adb logcat -v time -b all > locatlog.txt

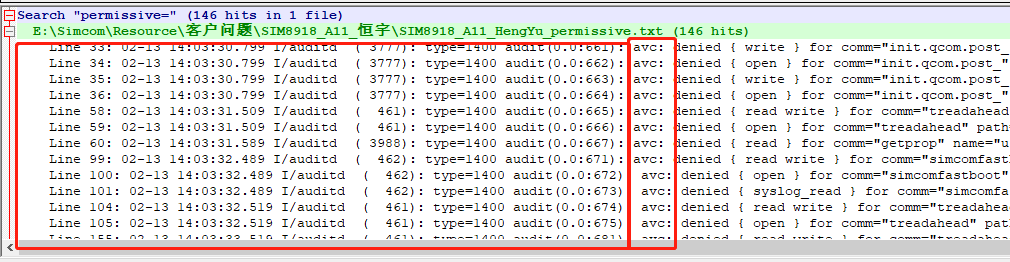
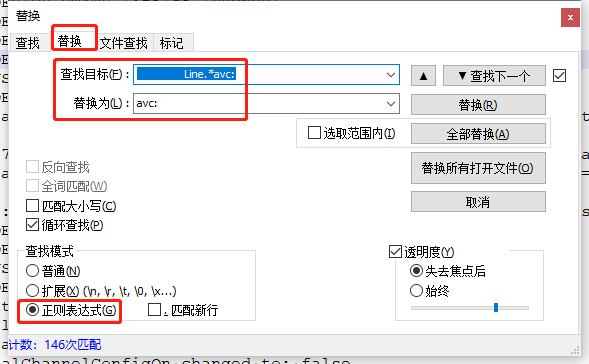
1. 将logcatlog.txt中avc相关的权限问题复制到te.txt中,如下所示：

IMG_256

提取步骤:

--locatlog.txt查找关键字“permissive=”  


--拷贝红框搜索内容到新的txt文件中

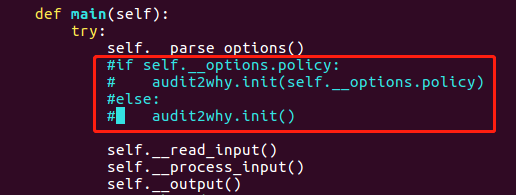
--根据正则表达式替换“ Line.\*avc:”内容为“avc:”  


1. 分析logcatlog.txt文件

audit2allow -i te.txt > avc.txt

打开avc.txt, 如果出现unable to open (null): Bad address 问题则需要作如下修改,如未出现则跳过以下修改：

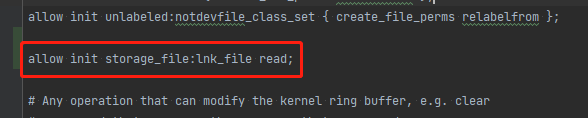
sudo vim /usr/bin/audit2allow 中注释掉以下红框中4句。



1. 打开avc.txt文件发现已生成te语句

allow init storage\_file:lnk\_file read;

1. 在源代码的system/sepolicy/public/init.te 文件中加入以上te语句：



相对应在system/sepolicy/prebuilts/api/XXX/public/init.te 中也需要加上以上语句。

XXX为最新api。 如有28、29、30，则选择30.

1. 重新编译系统即可。

# SELinux 常见错误分析

## 无SELinux权限

**错误如下：**

<5>[   22.281746] audit: type=1400 audit(3672.729:4): avc:  denied  { read } for  pid=251 comm="recovery" name="u:object\_r:sf\_lcd\_density\_prop:s0" dev="tmpfs" ino=2976 scontext=u:r:recovery:s0 tcontext=u:object\_r:sf\_lcd\_density\_prop:s0 tclass=file permissive=0

**解决方法：**

根据语法格式：allow scontext tcontext:tclass operation 得出：

allow recovery sf\_lcd\_density\_prop:file {read};

在scontext指定的te文件中即recovery.te中添加以上权限解决。

## 编译出错

遇到编译错误怎么办？（首先请排除拼写错误）说明此项权限是SELinux明确禁止的，也是Google CTS禁止的，如果产品不需要过CTS，可以修改。

一般来说，编译出错的log会提示相关哪个文件哪一行出错，文件位置一定会在对应的.te文件。比如domain.te规定了以下neverallow,

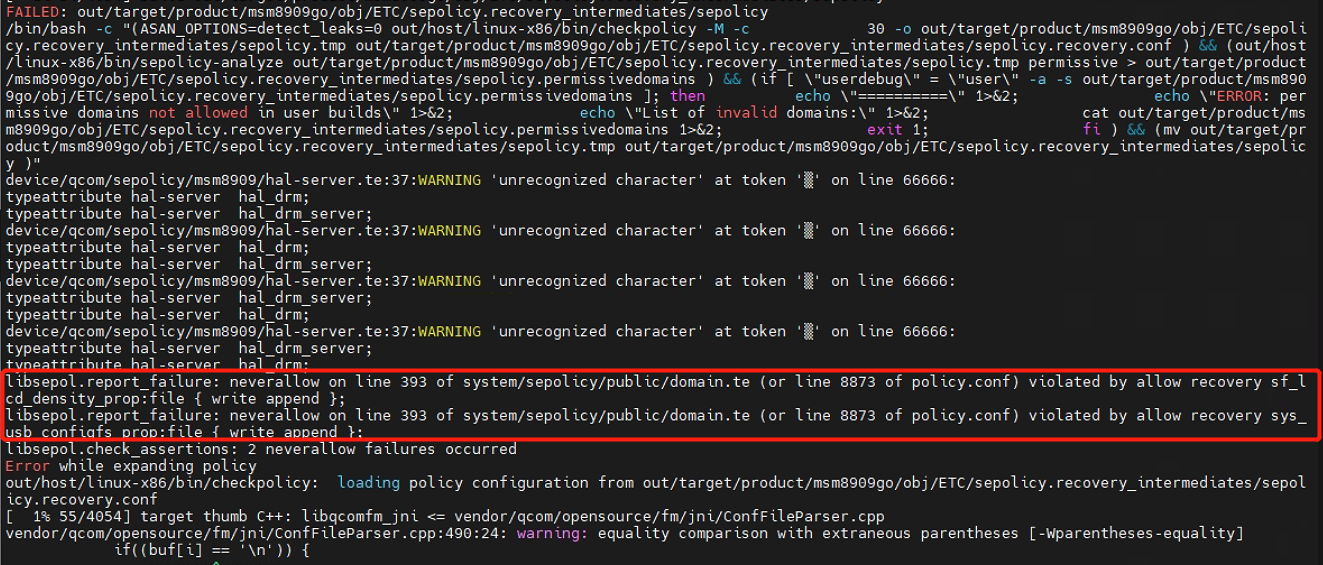
neverallow system\_server sdcard\_type:dir { open read write };

那么system\_server是不能拥有这些权限的，如果赋予这些权限就编译报错，解决方法是根据编译错误提示的行号，把这一句注释掉即可。

另外如有如下错误，则在对应地方排除掉。

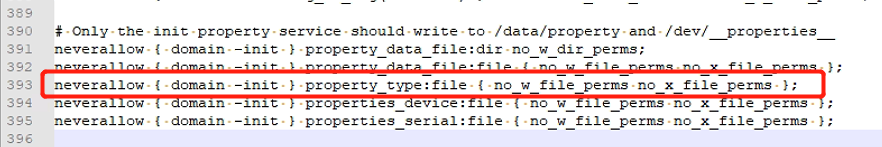
**错误如下**：

在recovery.te中增加相应的权限后编译，系统报错：



**解决方法：**

domain.te 文件中对应的地方如下：

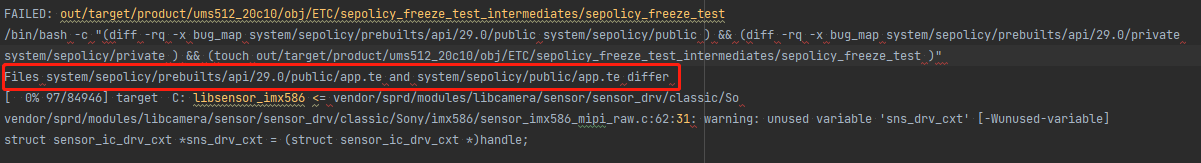


将neverallow中增加 recovery：

neverallow { domain –init -recovery} property\_type:file { no\_w\_file\_perms no\_x\_file\_perms};

## 修改的文件不一致

**错误如下：**



**解决方法：**

以上错误是因为system/sepolicy/prebuilts/api/29.0/public/app.te 和 system/sepolicy/public/app.te 文件不一致导致。将这两个文件修改为一致。

## 对应的te文件未找到

**错误如下：**

[ 14.546878] type=1400 audit(524743.979:9): avc: denied { net\_raw } for pid=566 comm="pm-service" capability=13 scontext=u:r:per\_mgr:s0 tcontext=u:r:per\_mgr:s0 tclass=capability permissive=0

对应的per\_mgr.te文件未找到。

**解决方法：**

1.在system/sepolicy/目录和device/qcom/目录或device/sprd/目录下搜索关键字”per\_mgr”;

2.如果能搜到该关键字，则在对应的文件中添加权限。

3.未搜到该关键字则在system/sepolicy/public/文件夹中创建per\_mgr.te文件, 定义type(可参考其它te文件中type的定义),然后将权限添加至该文件中。相应在system/sepolicy/prebuilts/api/XXX/public/中也创建该文件（XXX为最新API）。

## 其它细节

1. 有时候avc denied的log不是一次性暴露所有权限问题，要等解决一个权限问题之后，才会暴露另外一个权限问题。比如提示缺少某个目录的read权限，加入read之后，才显示缺少write权限，要一次试，一次次加，时间花费较多。

针对dir缺少的任何权限，建议赋予create\_dir\_perms，基本涵盖对dir的所有权限，比如：

{ open search write read rename create rmdir getattr }等等。

针对file缺少的任何权限，建议赋予rw\_file\_perms，基本涵盖对file的所有权限，比如：

包含{ open read write open execute getattr create ioctl }等等

2.当要加入的权限很多时，可以用中括号：

allow recovery vfat:dir { search write add\_name create};