|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 深圳市华曦达科技股份有限公司 | | 文 档 编 号 | 版本号 | 密级 |
| 文档编号 | V1.0 | 机密 |
| **文档名称** | SUD权限配置说明 | | 日期 | 2016-12-14 |

# SUD权限配置说明

dmc1

|  |  |  |  |
| --- | --- | --- | --- |
| **文档作者：** | **张定雄** | **日期：** | **2016-12-14** |
| **项目经理：** |  | **日期：** |  |
| **审 核：** |  | **日期：** |  |
| **批 准：** |  | **日期：** |  |

深圳市华曦达科技股份有限公司

文档历史发放及记录

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **变更（+/-）说明** | **作者** | **版本号** | **日期** | **批准** |
| 1 | 初稿 | 张定雄 | V1.0 | 2016-12-14 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

目 录

[SELinux策略配置文档 1](#_Toc469574240)

[文档的简要功能 3](#_Toc469574241)

[文档的适用范围 3](#_Toc469574242)

[一 SEAndroid配置 5](#_Toc469574243)

[2.1 背景知识 5](#_Toc469574244)

[2.2 SUD TE配置 5](#_Toc469574245)

[2.2.1 file\_contexts配置 5](#_Toc469574246)

[2.2.2 sud.te 配置 5](#_Toc469574247)

[2.2.3 安全策略配置说明 6](#_Toc469574248)

[2.2.4访问向量规则 7](#_Toc469574249)

[2.3 编译和打包 7](#_Toc469574250)

[2.3.1 编译 7](#_Toc469574251)

[2.3.2 打包 7](#_Toc469574252)

[2.4 验证 7](#_Toc469574253)

[2.5 SELinux打开和关闭 8](#_Toc469574254)

[2.5.1 打开SELinux 8](#_Toc469574255)

[reset 2.5.2 关闭SELinux 8](#_Toc469574256)

# 一 SEAndroid配置

## 2.1 背景知识

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。比如，以root用户启动Browser，那么Browser就有root用户的权限，在Linux系统上能干任何事情。由此Android用户可以通过一些系统漏洞，将自己开发的SU拷贝到/system/bin目录下面，这个SU具有Root权限的可执行程序，通过这一方法，实现一个普通的应用具有Root权限，这也是360超级Root应用的原理。

为了防止用户通过系统漏洞去获取Root权限，Google基于SELinux开发了一套新的权限控制，SEAndroid是从Android 4.4 开始增加的一套强制访问控制（Mandatory Access Control, MAC），在Android 本身的任意访问控制（Discretionary Access Control，DAC）之上，通过为对象增加安全上下文的方式，对访问的权限进行了精确的控制。

## 2.2 SUD TE配置

为了配置sud，需要增加一个新的te文件sud.te，存放位置为external/sepolicy目录，具体配置说明见下文。

### 2.2.1 file\_contexts配置

File\_contexts.te中保存的是系统中所有文件的安全上下文定义，init 进程将会根据file\_contexts来决定该专有目录和目录下的文件的安全标记。

由此我们定义如下字段：

/system/xbin/sud u:object\_r:sud\_exec:s0

意思为：为sud进程配置用户为u，角色为object\_r，type为sud\_exec，安全级别为s0。

### 2.2.2 sud.te 配置

Sud的te文件如下：

type sud\_exec, exec\_type, file\_type;

#说明：定义type为sud\_exec，并且sud\_exec在可执行类型和文件类型的安全上下文中，也就是sud\_exec要遵守exec\_type和file\_type的权限策略。

type sud, domain, mlstrustedsubject;

#定义type为sud，并且sud要遵循domain和mlstrustedsubject定义的权限策略

domain\_auto\_trans(init, sud\_exec, sud)

#domain\_auto\_trans是te\_macros文件定义的宏，具体策略见te\_macros

#在SELinux中需要进行DT(Domain Trans)转换，即某个进程的Domain切换到一个更合适的#Domain中去。

domain\_auto\_trans(dumpstate, sud\_exec, sud)

domain\_auto\_trans(sud, dumpstate\_exec, dumpstate)

permissive sud;

#sud采用permission模式

allow sud self:capability {net\_raw net\_admin dac\_override fsetid};

#具体的配置策略

allow sud system\_data\_file:dir {write add\_name rmdir remove\_name};

allow sud system\_data\_file:file {write create setattr unlink};

net\_domain(sud)

#设置sud遵循net域的安全策略

app\_domain(sud)

#设置sud遵循app域的安全策略

### 2.2.3 安全策略配置说明

在执行进程的过程中，如果存在权限不够的情况，将会出现如下的内核打印：

type=1400 audit(1481772449.302:3302): avc: denied { getattr } for pid=11395 comm="sh" path="/system/bin/toolbox" dev="mmcblk0p13" ino=346 scontext=u:r:sud:s0 tcontext=u:object\_r:toolbox\_exec:s0 tclass=file permissive=1

其中denied表示缺少的权限，scontext表示源上下文，tcontext表示目标上下文，tclass表示class

上下文: u:r:sud:s0 的格式为，用户：角色：类型：权限级别，策略仅和类型有关系

配置的安全权限策略为：

allow 源类型 目标类型:客体类别 许可

因此这一条的配置策略为：

allow sud toolbox\_exec:file getattr;

当有多个类似的权限的时候，可以配置为如下：

allow sud toolbox\_exec:file {getattr wirte read};

### 2.2.4访问向量规则

SELinux权限访问向量规则有以下四条：

1. allow：表示允许主体对客体执行允许的操作
2. dontaudit：表示不记录违反规则的决策信息，且违反规则不影响运行(允许操作且不记录)
3. auditallow：表示允许操作并记录访问决策信息(允许操作且记录)
4. neverallow： 表示不允许主体对客体执行指定的操作

但是注意neverallow规则，默认情况下所有的访问都是被拒绝的，设计这个规则的主要目的是为了帮助编写策略时，可以明确地指出不想要的访问许可，因此可以预防意外发生，回想一下，在一个SELinux策略中可能包含成千上万条规则，可能不小心加入了我们本不想授予的访问权，此时，neverallow规则就可以帮助预防这种情况发生。

## 2.3 编译和打包

### 2.3.1 编译

编译需要进入到SDK下面，编译目录为/external/sepolicy，执行命令：mm

编译生成文件目录：out\target\product\xxx\root

编译生成文件有：file\_contexts、property\_contexts、seapp\_contexts、sepolicy、service\_contexts、selinux\_version

编译的时候可能存在权限冲突问题，此时需要根据具体的情况看是否需要打开权限。

### 2.3.2 打包

打包需要替换root.tar.bz2文件中关于sepolicy相关的部分，步骤为：

1. 解压root.tar.bz2文件，命令为tar –xvf root.tar.bz2
2. 替换上文中提到的编译生成文件
3. 压缩root.tar.bz2文件，命令为tar –cjvf root.tar.bz2 root/
4. 执行打包命令

## 2.4 验证

验证分为两个部分：

1. 查看SELinux是否打开，查看命令为:getenforce，返回状态有三种可能性

Enforcing:表示SELinux已经打开

Permissive：表示SELinux关闭，但是打印警告，并不执行权限审查

Disable：表示SELinux关闭，且不打印警告

1. 确保SELinux打开的状态下，运行应用程序，查看应用程序是否执行成功

## 2.5 SELinux打开和关闭

### 2.5.1 打开SELinux

打开SELinux有两种方式：

方式一：setenforce Enforcing，这种方式只是在运行过程中打开SELinux，一旦重启机顶盒，STB的SELinux状态恢复为设置之前的状态

方式二：在uboot下面打开SELinux，在STB进入Uboot模式下面，输入下面的命令打开：

setenv storeargs 'setenv bootargs rootfstype=ramfs init=/init console=ttyS0,115200 no\_console\_suspend earlyprintk=aml-uart,0xc81004c0 androidboot.selinux=Enforcing logo=${display\_layer},loaded,${fb\_addr} vout=${outputmode},enable osd\_reverse=${osd\_reverse} video\_reverse=${video\_reverse} jtag=${jtag} androidboot.firstboot=${firstboot}; run cmdline\_keys;'

saveenv

### reset 2.5.2 关闭SELinux

和打开SELinux的方式类似：

方式一：setenforce disabled

方式二：uboot下执行命令：

setenv storeargs 'setenv bootargs rootfstype=ramfs init=/init console=ttyS0,115200 no\_console\_suspend earlyprintk=aml-uart,0xc81004c0 androidboot.selinux=disabled logo=${display\_layer},loaded,${fb\_addr} vout=${outputmode},enable osd\_reverse=${osd\_reverse} video\_reverse=${video\_reverse} jtag=${jtag} androidboot.firstboot=${firstboot}; run cmdline\_keys;'

saveenv

reset