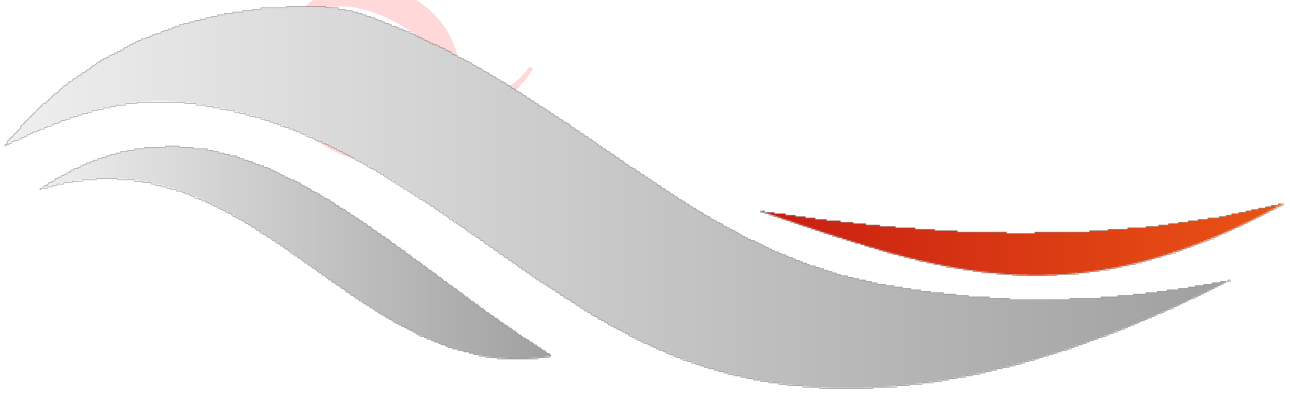
SElinux 使用文档

Author **Darren**

Rivision **V0.1**

Date **2017/11/23**



Our aim is to provide customers with timely and comprehensive service. For any assistance, please contact our company headquarters:

Quectel Wireless Solutions Co., Ltd.
Office 501, Building 13, No.99, Tianzhou Road, Shanghai, China, 200233
Tel: +86 21 5108 6236
Email: info@quectel.com

Or our local office. For more information, please visit:
http://www.quectel.com/support/salesupport.aspx

For technical support, or to report documentation errors, please visit:
http://www.quectel.com/support/techsupport.aspx
Or Email to: Support@quectel.com

GENERAL NOTES

QUECTEL OFFERS THE INFORMATION AS A SERVICE TO ITS CUSTOMERS. THE INFORMATION PROVIDED IS BASED UPON CUSTOMERS' REQUIREMENTS. QUECTEL MAKES EVERY EFFORT TO ENSURE THE QUALITY OF THE INFORMATION IT MAKES AVAILABLE. QUECTEL DOES NOT MAKE ANY WARRANTY AS TO THE INFORMATION CONTAINED HEREIN, AND DOES NOT ACCEPT ANY LIABILITY FOR ANY INJURY, LOSS OR DAMAGE OF ANY KIND INCURRED BY USE OF OR RELIANCE UPON THE INFORMATION. ALL INFORMATION SUPPLIED HEREIN IS SUBJECT TO CHANGE WITHOUT PRIOR NOTICE.

COPYRIGHT

About the Document

History

Revision	Date	Author	Description
0.1	November 23,2017	Darren	Initial

目 录

第 1 章　SElinux 使用方法

SElinux 是一个强制访问控制系统，它为每个进程与文件都打上一个安全上下文标签，而 selinux 通过这个标签对系统访问控制进行管理。所以使用 selinux 之前需要确保下面几项：

- 系统配置：在/etc/selinux/config 文件中必须初始化 selinux 工作状态
- 文件标签：每个文件都需要有特定的安全上下文标签，没有标签的文件，selinux 无法管理
- 策略文件：在系统/etc/selinux/quectel/policy 目录下必须有 selinux 的策略文件

只要满足以上三项，selinux 就可以正常工作了。系统配置如下：

/etc/selinux/config

```
# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
# enforcing - SELinux security policy is enforced.
# permissive - SELinux prints warnings instead of enforcing.
# disabled - No SELinux policy is loaded.
SELINUX=permissive
# SELINUXTYPE= can take one of these two values:
# default - equivalent to the old strict and targeted policies
# mls     - Multi-Level Security (for military and educational use)
# src     - Custom policy built from source
SELINUXTYPE=quectel

# SETLOCALDEFS= Check local definition changes
SETLOCALDEFS=0
```

配置文件如果不存在，selinux 是无法启动的。SELINUX 环境变量的作用是初始化 selinux 工作状态，permissive 是宽容模式，是供调试使用的。enforcing 是强制模式，在这种模式下，selinux 会拒绝一切不满足安全策略的系统行为。SELINUXTYPE 环境变量决定了 selinux 加载策略时，会去哪个目录寻找 policy 文件，当设置为 quectel 之后，selinux 会去/etc/selinux/quectel/policy 目录下寻找策略文件

第 2 章　selinux 文件标签设置

制作镜像的步骤与其它项目完全相同，唯一的区别就是，我们需要使用特别制作的 mkfs.ubifs，这个修改的工具是支持打 selinux 标签的。操作方法就是在 mkfs.ubifs 命令中加上 --selinux=file_contexs，这个 file_contexts 是标签配置文件，需要我们自己手动制作。

```
darren@darren:~/sdb/system$sudo ./mkfs.ubifs -m 4096 -e 253952 -c 2146 -r rootfs/ -F
-o rootfs.ubifs --selinux=file_contexts
darren@darren:~/sdb/system$sudo ubinize -o system.ubi -p 256KiB -m 4096 -s 4096
ubinize_system_rootfs.cfg
```

2.1　file_contexts

file_contexts 就是 selinux 所有文件安全标签的配置文件（这里的所有是包括镜像文件与系统动态创建的文件），所以 file_contexts 不仅是在 mkfs.ubifs 进需要使用，也需要把这个文件导入到/etc/selinux/目录一下，导入到模块中的原因是系统动态创建的文件也要有安全标签的来源。file_contexts 语法如下：

```
# Syntax of file context description

regexp <-type> ( <file_label> | <<none>> )

regexp：这个是正则表达式
file_label:安全上下文字符串
<<none>>:忽略不设置

-type类型：
--：只匹配普通文件
-d：只匹配目录
-l：只匹配符号链接文件
-c：只匹配字符设备文件
-b：只匹配块设备文件
不指定类型：匹配任何文件
```

file_contexts 文件的编写语法就是这么多。下面看几个例子，并一一解释：

```
/.*         u:object_r:rootfs:s0
/bin/   -d  u:object_r:bin:s0
/bin/.*     u:object_r:bin_exec:s0
/bin/sh -l  u:object_r:shell_exec:s0

file_contexts会逐一匹配，当设置/text.txt文件时，它匹配/.*是成功的，但匹配后面都会失败，所以/text.txt
安全上下文就会被设置为u:object_r:rootfs:s0，当设置/bin/dmesg文件时，它匹配/.*是成功的，匹配/bin/.*
也是成功的，后面的都会匹配失败，所以它会被设置为/bin/.*的安全上下文，也就是u:object_r:bin_exec:s0
```

当编写 file_contexts 文件时，要设置的安全上下文需要与后面讲的策略相一致。也就是 file_contexts 设置的安全上下文是策略定义的，否则无意义。

第 3 章　SElinux 策略编写

SElinux 策略包括了，客体类别和权限集，用户，角色，类型，多级权限，条件控制等，SElinux 策略文件框架如下所示：

```
sepolicy/
├── fs_use.................. 定义文件系统的扩展属性，如果不定义，文件系统则不能设置安全上下文
├── genfs_contexts......... 设置文件系统中文件默认安全上下文，动态设置 proc 文件系统安全上下文
├── macros/
│   ├── global_macros..................................................... 全局宏，定义客体类别权限组
│   ├── te_macros.......................................................... 策略文件编写需要的宏
│   └── mls_macros.......................................................... 多级权限宏定义
├── users/
│   └── users.............................................................. 定义 selinux 用户
├── roles/
│   └── roles.............................................................. 定义 selinux 角色
├── type/
│   ├── file_type.......................................................... 定义文件的 selinux 类型
│   └── attributes......................................................... 定义 selinux 的类型属性
├── class/
│   ├── classes............................................................ 定义 selinux 客体类别
│   └── perms.............................................. 定义 selinux 客体类别权限集
├── mls/
│   └── mls................................................................ 定义 selinux 多级权限策略
├── sid/
│   ├── initial_sids....................................................... 定义缺省安全上下文序列
│   └── initial_sid_contexts.............................................. 初始化缺省安全上下文
├── app/
│   ├── system/.......................................... 该目录下的文件是定义系统进程的权限策略
│   │   ├── kernel.te..................................... 定义 kernel 类型进程的权限策略
│   │   ├── init.te....................................... 定义 init 类型进程的权限策略
│   │   └── shell.te...................................... 定义 shell 类型的进程权限策略
│   └── quectel/......................................... 该目录下的文件是定义移远进程的权限策略
│       ├── quectel-remotefs-service.te.................... 定义 remotefs 进程权限策略
│       └── quectel_daemon.te.............................. 定义 quectel_daemon 进程权限策略
└── Makefile.................................................................. 编译策略文件
```

selinux 策略编写基础，首先 selinux 控制的主体是进程，它限制进程对资源的访问权限。那么客体就是资源，包括文件，设备，网络，信号，进程等多种客体。每个主体与客体都必须有一个安全上下文，这个安全上下文就决定了主体进程对客体资源是否有访问权限。

3.1　客体类别

什么是客体，客体就是主体要访问的东西，客体可以是文件，可以是字符设备，可以 socket 文件等，客体定义在 sepolicy/class/classes 中定义。那么什么是客体权限集，客体不同，权限也不同，比如，如何客体是文件，那么文件有创建 / 打开 / 读 / 写 / 执行等操作，如果客体是 socket，那么客体操作集有 bind / listen / accept / connect 等。这些移远已经全部定义好了，不需要修改，也不需要添加。客体类别定义语法如下：

```
#类别定义语法
class class_name
class：类别定义关键字
class_name：要定义的类别标识符(可以是字母，数字，下划线)
```

```
示例：
class file
class dir
class chr_file

#类别操作集合语法
common common_name {perm_set}
common：操作集合定义关键字
common_nam：操作集合标识符
perm_set：操作集
示例：
common file_perm
{
    read
    write
    ioctl
    ...
}

#客体类别与操作关联
class class_name [inherits common_name] [{perm_set}]
class：关键字，这里不是要定义一个新类别，而是把前面定义的类别与操作
inherits：关键字，代表一个类别继承一个操作集合
perm_set：继承的操作集合如果不够，还可以添加新的操作
示例：
class chr_file
inherits file_perm
{
    execute_no_trans
    entrypoint
    execmod
    open
    audit_access
}
```

3.2 用户与角色

file_contests 中有这样一个安全上下文"u:object_r:bin_exec:s0"，其中 u 就是用户，object_r 是角色，bin_exec 是类型，s0 是多级安全。在 SElinux 中最主要的访问控制是 TE 控制，也就是类型增强控制，对于用户与角色关注不高，用户，角色已经定义，不需要修改。语法如下：

```
#SElinux角色定义
role role_name; 或
role role_name [types type_set];
role：角色定义关键字
role_name：角色标识符
第一种是直接定义一个角色，第二种是定义角色的同时，关联一个类型
示例：
role r;
role r types domain;#如果前面定义了r角色，这里只要关联一个domain类型就可以了

#SElinux用户定义
user user_name roles {role_set};
user：用户定义关键字
user_name：用户标识符
roles：与角色关联的关键字
role_set：关联的角色集
示例：
user u roles {r};
```

3.3 SElinux 类型

前面的用户与角色不是我们关注的重点，基本上不需要修改它们。这里最重要的就是 TE 控制，而 TE 控制就是类型增强控制。所以类型是需要特别关注。类型是安全上下文中的第三个成员，"u:object_r:bin_exec:s0" 中的 bin_exec 就是类型。假如我们在策略中写入"allow bin_exec data_t:file open read" 这就代表 bin_exec 类型的进程对 data_t 类型的文件有 open 和 read 操作的权限。所以类型才是我们最关注的，语法如下：

```
#SElinux类型定义语法
type type_name [ alias alias_set ] [, attribute_set] ;
type：类型定义关键字
type_name：类型标识符
alisa：定义类型别名的关键字
alisa_set：别名集合
attribute_set：类型属性集合（属性与类型标识符处在同一命名空间，也就是属性与类型不能重名）
示例：
type bin_exec,file_type,exec_type;#定义一个bin_exec类型，有file_type与exec_type两个属性

#SElinux类型属性定义
attribute attribute_name;
attribute：属性定义的关键字
attribute_name：属性标识符
注意：属性本质也是类型，可以说它是类型的类型，它类似于一个数组，这个数组中有多个类型。

#类型与属性关联语法
typeattribute type_name attritbute_name;
注意：当我们定义一个新的类型时没有关联某个属性，那么就可以通过typeattribute来关联
示例：
type user_file_t;
attribute file_type;
type system_file_t,file_type;#定义类型时直接关联
typeattribute user_file_t file_type;#定义类型完成后关联

属性的作用：
allow bin_exec file_type:file {open read};
这个策略语句可以说明属性的作用，这个语句等价于下面两个语句的组合
allow bin_exec user_file_t:file {open read};
allow bin_exec system_file_t:file {open read};
```

3.4 安全上下文转换

对于 TE 控制来说，如果 selinux 不能进行上下文转换，那么 selinux 是没有任何作用的。selinux 是对主体进程访问客体资源进行安全控制，那么主体进程是随机创建的，比如 init 进程可以 fork 一个 shell 进程出来，如果 shell 进程与 init 进程保持相同的安全上下文，那 selinux 就没了控制的意义。所以 shell 进程的安全上下文必须与 init 进程的不同。这里就涉及到了安全上下文转换。

安全上下文转换是在 execve 函数簇调用时完成的，也就是 execve 加载可执行文件时，会通过 selinux 策略去检查是否要进行安全上下文转换，如果需要则转换。比如，init 进程的类型为 init_t，shell 的可执行文件为/bin/sh，类型为 shell_exec，现在希望 init 进程启动一个 shell 时，shell 进程类型为 shell_t，那么至少需要满足下面几点：

- 执行权限：init 进程必须对/bin/sh 文件有执行权限 (这个显而易见)
- entrypoint：shell_t 类型必须对/bin/sh 文件有 entrypoint 权限
- transition：init_t 类型进程必须被允许向 shell_t 类型进程转换权限

```
#进程安全上下文转换语法
type_transition source_type temp_type : class target_type
type_transition：类型转换关键字
source_type：原类型
target_type：目标类型
temp_type：对于进程转换来说就是文件类型
示例：
type_transition init_t shell_exec:process shell_t;
allow init_t shell_exec:file { getattr open read execute };#满足第一个执行权限条件
allow shell_t shell_exec:file { entrypoint open read execute getattr };#满足第二个entrypoint权限
allow init_t shell_t:process transition;#满足第三个transition权限
注意：对于这种语句比较多的策略定义，移远定义了宏操作，可以一步完成，例如：
domain_auto_trans(init_t,shell_exec,shell_t)
```

第 4 章　实际操作例子

策略的框架已经完成，用户，角色，多级访问等都不需要关注，假如现在有一个程序 test_dameon.out，这是一个可执行程序，文件类型是 test_daemon_exec。现在需要在策略中限制它只对类型为 data_file_type 的文件有访问权限，那么策略写法是：创建一个 test_daemon.te 文件，然后把文件放在 sepolicy/app/quectel/目录下面：

sepolicy/app/quectel/test_daemon.te

```
#创建test_daemon类型，这是test_dameon.out成功运行后进程的类型
type test_daemon,domain,mlstrustedsubject;
#创建test_daemon.out的文件类型为test_daemon_exec，这个还需要在file_contexts文件中写上下面句子：
#/bin/test_daemon.out -- u:object_r:test_daemon_exec:s0
type test_daemon_exec, exec_type, file_type;

init_daemon_domain(test_daemon)

allow test_daemon data_file_type:file {open read};
```

如果有进程 selinux 权限拒绝而不能运行，那么在内核 log 中获取到 avc log，示例如下：

```
[ 9960.257215] audit: type=1400 audit(315974762.483:473): avc: denied { read } for pid=1263 comm="sh" name="87" dev="proc" ino=6611 scontext=u:r:init:s0 tcontext=u:r:kernel:s0 tclass=dir permissive=1
```

selinux 拒绝 log 的关键字已经标记为红色了，scontext 是主体进程的安全上下文，tcontext 是客体资源的安全上下文，tclass 是客体类别，denied read 中被拒绝的操作。

所以翻译过来就是 init 类型的进程 (scontext 决定)，对 kenel 类型 (tcontext 决定) 的目录 (tclass 决定) 缺少 read(denied 决定) 权限，所以策略可以这么写：allow init kernel:dir read;