Android安全机制及SEAndroid

黎鹏

目录

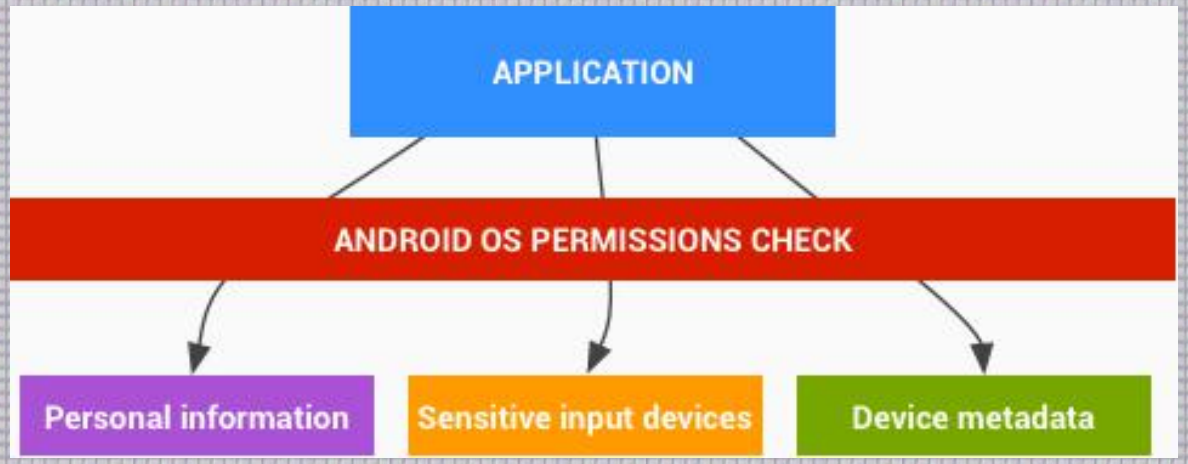
- 简要了解Permission机制
- 较为深入介绍Android DAC机制: Android IDs，capabilities
- 着重介绍SEAndroid机制

Android系统的安全性

- 在Linux Kernel层应用各种安全机制, DAC和SELinux等;
- 所有的应用程序都被强制运行在自己的Sandbox中;
- 严格的进程间通信IPC机制
- 应用程序签名
- Permission机制
- …

Android安全-----Permission机制

- Android 上的所有应用均在应用沙盒内运行。默认情况下，Android的应用只能访问有限的系统资源. 对于那些很可能影响用户体验, 及网络, 数据, 费用相关操作, 系统都会做出严格控制.



Permission机制用来授予权限.

- 摄像头功能
- 位置数据 (GPS)
- 蓝牙功能
- 电话功能
- 短信/彩信功能
- 网络/数据连接
- ...

Android安全-----Permission机制

- Android应用程序在默认时没有访问资源或数据的权利, 需要显性地作出Permission申请.
- Android M以后版本发生了一些变化, 参见Runtime Permission.
- 主要是通过AndroidManifest.xml中的<uses-permission>标签来完成.

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="net.suntec.merbok.media.test.provider"
    android:sharedUserId="android.uid.system">
    <uses-permission android:name="android.permission.MANAGE_ACTIVITY_STACKS"/>
    <uses-permission android:name="android.permission.REORDER_TASKS"/>
```

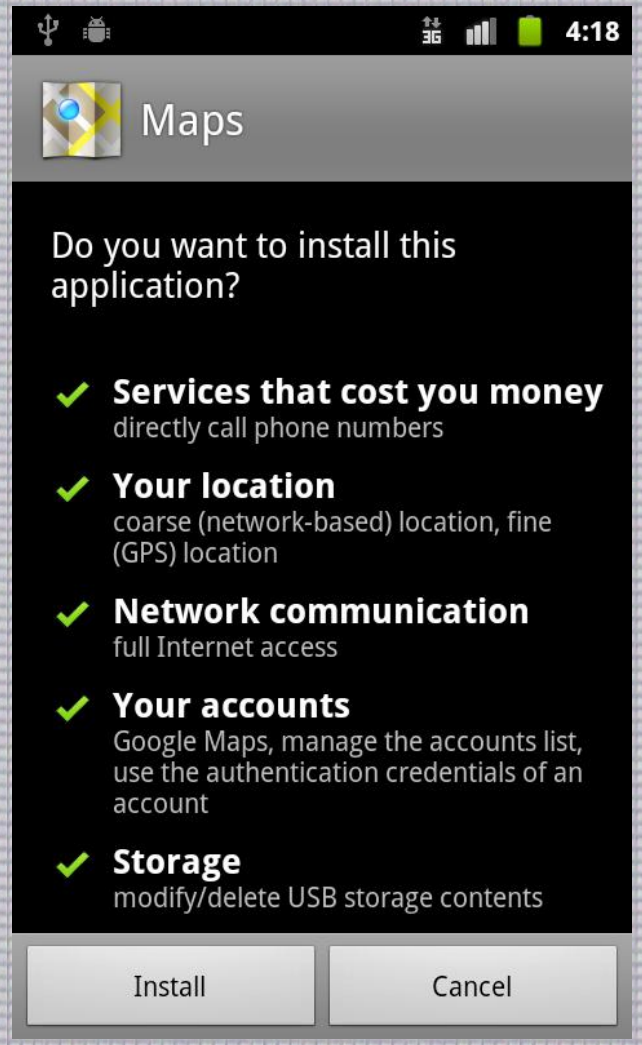
Android安全-----Permission机制

- 常见权限:

RECEIVE_SMS	Allows an application to receive SMS messages.
CALL_PHONE	Allows an application to initiate a phone call
BATTERY_STATS	Allows an application to collect battery statistics
CAMERA	Required to be able to access the camera device.
…	…

完整权限可参考:

- https://developer.android.com/reference/android/Manifest.permission.html

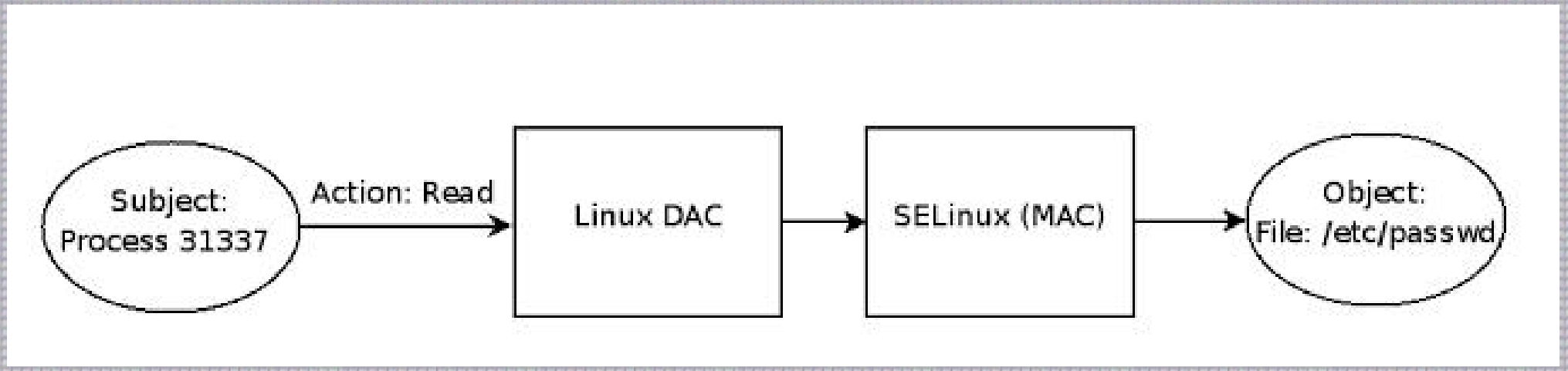


我们代码里: 常见需要权限的操作

- system("/system/bin/mdnsd &")
- property_set("ctl.start", MDNS_SERVICE_NAME);
- 文件操作
- I2C操作
- binder
- 配置网络system("ip -6 route XXX")
- connect, socket等
- ...

Android的DAC和MAC安全机制

- DAC (Discretionary Access Control)
  自主访问控制
  . 9位权限码(User-Group-Other),
  . 访问控制列表ACL
- MAC(Mandatory Access Control)
  强制访问控制
  . MultiLevel Secure, MLS,
  . SElinux/SEAndroid:
  Security-Enhanced Linux/Android



Android安全-----DAC保护

UGO传统的安全保障手段, 虽然中规中矩, 但依然起到一定的保护作用.

user group other r w x UID GID SUID SGID

```
root@lipeng-pc:/bin# ll ping
-rwsr-xr-x 1 root root 44168  5月  8  2014 ping*
root@lipeng-pc:/bin# ll
total 10584
drwxr-xr-x  2 root root     4096  8月  3  2017 ./
drwxr-xr-x 25 root root     4096  8月  3  2017 ../
-rwxr-xr-x  1 root root  1021112  5月 16  2017 bash*
-rwxr-xr-x  1 root root    31152 10月 21  2013 bunzip2*
-rwxr-xr-x  1 root root  1918032 11月 15  2013 busybox*
-rwxr-xr-x  1 root root    31152 10月 21  2013 bzcat*
lrwxrwxrwx  1 root root        6  6月 26  2017 bzcmp -> bzdiff*
```

Android安全-----DAC保护

- id 命令

```
root@lipeng-pc:/bin# id
uid=0(root) gid=0(root) groups=0(root)
```

```
lipeng:bin $ id
uid=1000(lipeng) gid=1000(lipeng) groups=100
0(lipeng),4(adm),24(cdrom),27(sudo),30(dip),
46(plugdev),108(lpadmin),124(sambashare)
```

Android安全-----DAC保护

- Linux是一个多用户系统, 而Android则不存在"用户登录"的过程, 那么它是如何区分"User"的呢?
- Android中运行的app进程UID和GID是由PackageManagerService分配的, 而且时在APK安装之初就确定下来的.
- 那么Service进程和文件的DAC配置呢?
- Android IDs:

(croot)/system/core/include/private/android_filesystem_config.h

```
#define AID_ROOT             0  /* traditional unix root user */
#define AID_SYSTEM        1000  /* system server */
#define AID_RADIO         1001  /* telephony subsystem, RIL */
#define AID_BLUETOOTH     1002  /* bluetooth subsystem */
...
```

Android安全-----DAC保护

- DAC中的capabilities机制
- 传统的Unix将进程分为特权进程(uid = 0, 如superuser或root)和非特权进程(uid !=0), 特权进程可以越过任何permission check, 而非特权进程需要根据UID, GID和groups进行全部的permission check.
- 从kernel 2.2之后, 将根特权划分成更小的特权，所以可以只用根用户特权的一个子集来运行任务. 可以将能力(capabilities)分配给文件.
- Capabilities list:
- http://man7.org/linux/man-pages/man7/capabilities.7.html

Android安全-----DAC保护

- Thread capability sets:
- 进程有三个能力集：
- 允许（permitted，P)
- 可继承（inheritable，I）
- 有效（effective，E）

- https://www.ibm.com/developerworks/cn/linux/l-posixcap.html?ca=drs-cn

Android安全-----DAC保护

- POSIX 能力将根特权划分成更小的特权，所以可以只用根用户特权的一个子集来运行任务。文件能力特性可以给一个程序分配这样的特权，这大大简化了能力的使用。在 Linux 中已经可以使用 POSIX 能力了。与将用户切换为根用户相比，使用能力有几个好处：
- (1) 可以将能力从有效集（effective set）中删除，但是保留在允许集（permitted set）中，从而防止滥用能力。
- (2) 可以从允许集中删除所有不需要的能力，这样就无法恢复这些能力。坦率地说，大多数能力是危险的，可能被滥用，减少攻击者可以利用的能力有助于保护系统。
- (3) 在对常规的可执行文件执行 exec(3) 之后，所有能力都会丢失

Android安全-----DAC保护

/external/libcap

- 查看进程能力
- getpcaps

```
root@lipeng-pc:/home/lipeng# ps -e | grep ping
17350 pts/27   00:00:00 ping
root@lipeng-pc:/home/lipeng# getpcaps 17350
Capabilities for `17350': = cap_net_admin,cap_net_raw+p
```

- 查看文件能力

setcap
getcap

```
root@lipeng-pc:/bin# setcap cap_net_raw=p ping
root@lipeng-pc:/bin# getcap ping
ping = cap_net_raw+p
root@lipeng-pc:/bin# setcap cap_net_raw=ep ping
root@lipeng-pc:/bin# getcap ping
ping = cap_net_raw+ep
```

Android安全-----DAC保护

- 例子: 利用文件能力特性将能力分配给程序

```
root@lipeng-pc:/bin# ll ping
-rwsr-xr-x 1 root root 44168  5月   8  2014 ping*
```

```
lipeng:~ $ ping www.baidu.com
PING www.a.shifen.com (115.239.210.27) 56(84) bytes of data.
64 bytes from 115.239.210.27: icmp_seq=1 ttl=49 time=7.36 ms
64 bytes from 115.239.210.27: icmp_seq=2 ttl=49 time=6.34 ms
^C
```

```
root@lipeng-pc:/bin# chmod u-s ping
root@lipeng-pc:/bin# ll ping
-rwxr-xr-x 1 root root 44168  5月   8  2014 ping*
```

```
lipeng:~ $ ping www.baidu.com
ping: icmp open socket: Operation not permitted
```

```
root@lipeng-pc:/bin# getcap ping
root@lipeng-pc:/bin# setcap cap_net_raw=ep ping
root@lipeng-pc:/bin# getcap ping
ping = cap_net_raw+ep
```

```
lipeng:~ $ ping www.baidu.com
PING www.a.shifen.com (115.239.210.27) 56(84) bytes of data.
64 bytes from 115.239.210.27: icmp_seq=2 ttl=49 time=5.98 ms
64 bytes from 115.239.210.27: icmp_seq=3 ttl=49 time=5.84 ms
64 bytes from 115.239.210.27: icmp_seq=4 ttl=49 time=5.71 ms
^C
```

难点:
不知道程序需要什么能力?
调试的问题?(strace及其它)

Android安全-----DAC保护

- (一)　Service进程的DAC配置: 在init.<xxx>.rc中

ep: service carplay /system/bin/xxxx

class core

user xxx

group xxx

1) user <username>

Change to ‘username’ before exec'ing this service. Currently defaults to root. (??? probably should default to nobody) As of Android M, processes should use this option even if they require Linux capabilities. Previously, to acquire Linux capabilities, a process would need to run as root, request the capabilities, then drop to its desired uid. There is a new mechanism through fs_config that allows device manufacturers to add Linux capabilities to specific binaries on a file system that should be used instead. This mechanism is described on http://source.android.com/devices/tech/config/filesystem.html. When using this new mechanism, processes can use the user option to select their desired uid without ever running as root. As of Android O, processes can also request capabilities directly in their .rc files. See the “capabilities” option below.

https://android.googlesource.com/platform/system/core/+/master/init/README.md

Android安全-----DAC保护

- 文件 DAC 配置

  http://source.android.com/devices/tech/config/filesystem.html

Android安全-----SEAndroid

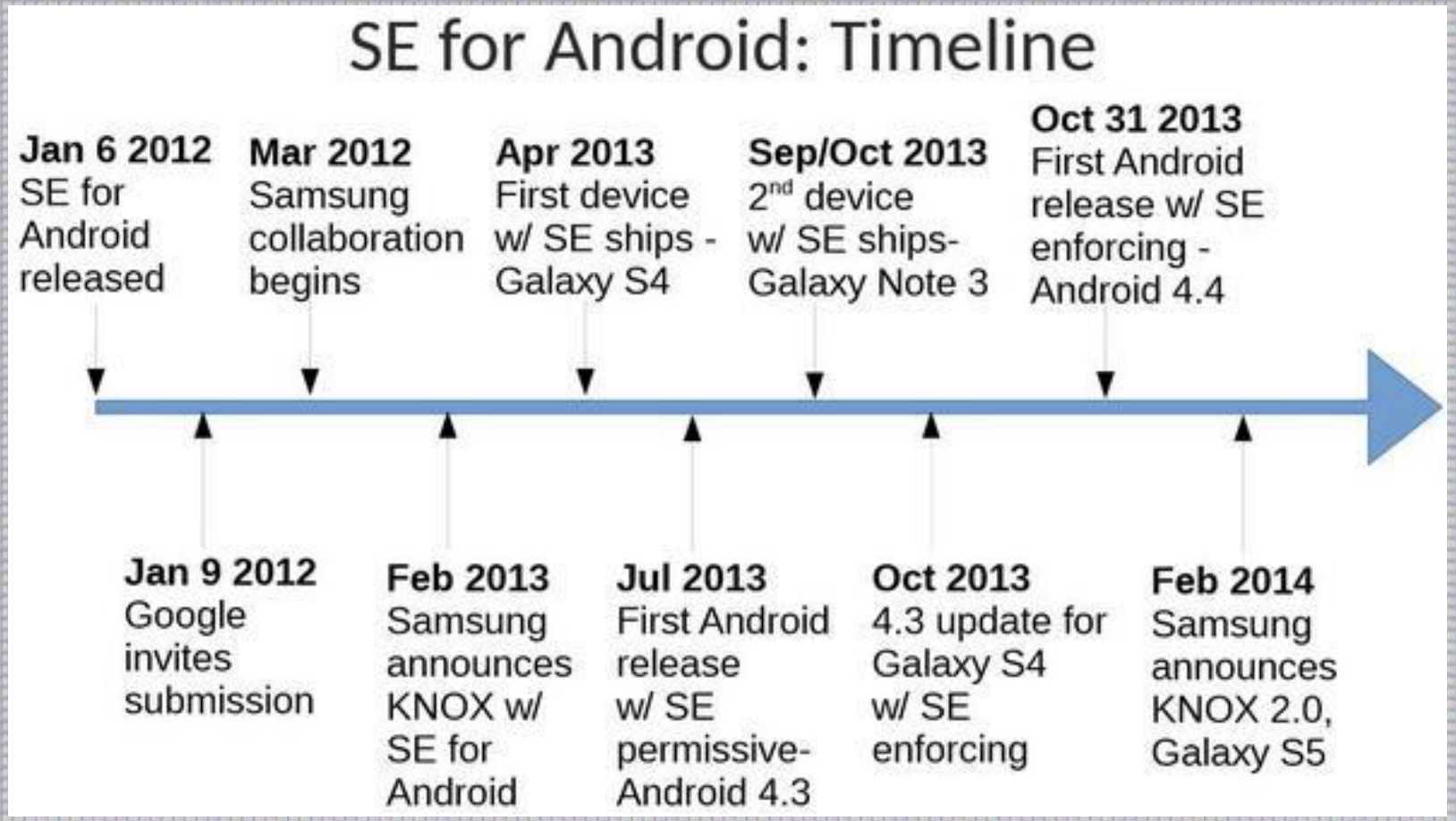
- 作为 Android 安全模型的一部分，Android 使用 SELinux 对所有进程强制执行强制访问控制 (MAC)，其中包括以 Root/超级用户权限运行的进程（也称为 Linux 功能）。SELinux 能够限制特权进程并能够自动创建安全政策，从而可提升 Android 的安全性。

- 借助 SELinux，Android 可以更好地保护和限制系统服务、控制对应用数据和系统日志的访问、降低恶意软件的影响，并保护用户免遭移动设备上的代码可能存在的缺陷的影响。

- Android 中包含 SELinux（处于强制模式）和默认适用于整个 AOSP 的相应安全政策。在强制模式下，非法操作会被阻止，并且尝试进行的所有违规行为都会被内核记录到 dmesg 和 logcat 中。Android 设备制造商应收集与错误相关的信息，以便在实施其软件和 SELinux 政策之前先对其进行优化

https://source.android.com/security/selinux/?hl=zh-cn

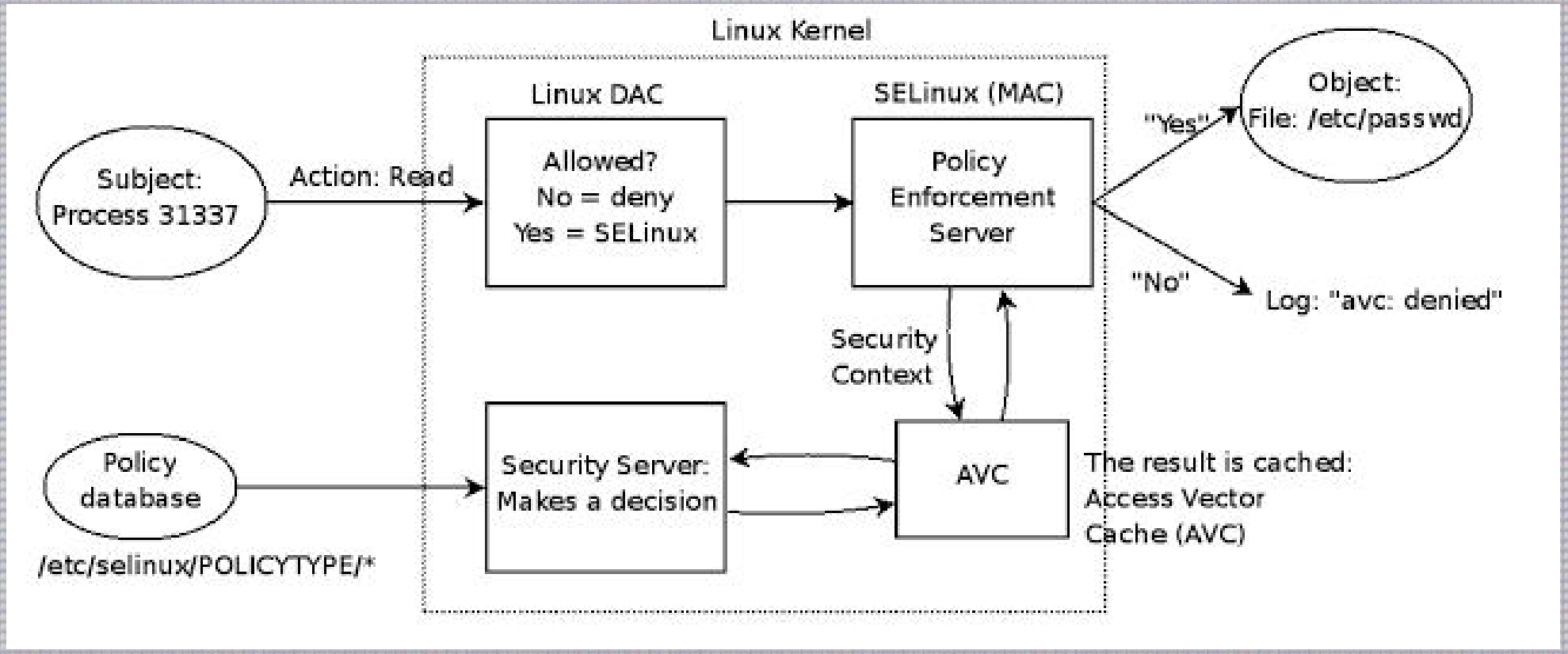
Android安全-----SEAndroid



Android安全-----SEAndroid

SELinux 可按两种全局模式之一运行：

(1)宽容模式(Permissive)：权限拒绝事件会被记录下来，但不会被强制执行。
(2)强制模式(Enforing)：权限拒绝事件会被记录下来并强制执行。



Android安全-----SEAndroid

- **Android 7 Selinux安全主要目录和文件**
- build/tools/fs_config
- external/libselinux/
- external/selinux/
- external/libcap/progs/
- device/xxx/xxx/sepolicy/
- device/xx/BoardConfig.mk
- system/sepolicy/
- system/core/libcutils/fs_config.c
- system/core/include/private/

  `---android_filesystem_capability.h

  `---android_filesystem_config.h

```
+-- build
|   `-- tools
|       `-- fs_config
+-- device
|   +-- generic
|   |   `-- brillo
|   |       `-- sepolicy
|   `-- hzak
|       `-- rpi3b
|           +-- base
|           |   `-- sepolicy
|           `-- fs_config
|               `-- android_filesystem_config.h
+-- external
|   +-- libcap
|   |   `-- progs
|   |       +-- getcap.c
|   |       +-- getpcap.c
|   |       `-- setcap.c
|   +-- libcap-ng
|   +-- libselinux
|   |   `-- src
|   |       `-- android.c
|   +-- minijail
|   +-- selinux
|   `-- sepolicy
`-- system
    +-- core
    |   +-- include
    |   |   `-- private
    |   |       +-- android_filesystem_capability.h
    |   |       `-- android_filesystem_config.h
    |   `-- libcutil
    |       `-- fs_config.c
    `-- extra
        `-- ext4_utils
            `-- contents.c
```

Android安全-----SEAndroid

常用相关命令:

- getenforce
- setenforce
- 通过ls -Z 命令查看系统中文件的SELinux security context
- 通过ps -Z命令查看当前系统进程的SELinux security context

```
$ adb shell ls -Z /system/bin
u:object_r:system_file:s0              acpi
u:object_r:apmanager_exec:s0           apmanager
u:object_r:system_file:s0              audio_hal_playback_test
u:object_r:system_file:s0              audio_hal_record_test
u:object_r:system_file:s0              avahi-browse
u:object_r:avahi_exec:s0               avahi-daemon
u:object_r:system_file:s0              base64
u:object_r:system_file:s0              basename
...
```

```
$ adb shell ps -Z
LABEL                          USER      PID   PPID  VSIZE  RSS     WCHAN              PC  NAME
u:r:init:s0                    root      1     0     6672   1508    SyS_epoll_ 00089cec S /init
u:r:kernel:s0                  root      2     0     0      0         kthreadd 00000000 S kthreadd
...
u:r:weaved:s0                  system    122   1     13304  7484    SyS_epoll_ 76b811c0 S /system/bin/weaved
u:r:webservd:s0                webserv   123   1     10592  6436    SyS_epoll_ 768431c0 S /system/bin/webservd
u:r:apmanager:s0               system    129   1     6864   4584    SyS_epoll_ 76b5c1c0 S /system/bin/apmanager
u:r:shill:s0                   root      131   1     10460  6684    SyS_epoll_ 76a161c0 S /system/bin/shill
u:r:tlsdated:s0                root      138   117   4744   368      pipe_wait 76ea9354 S /system/bin/tlsdated
```

Android安全-----SEAndroid

- 通过id命令查看当前shell的uid, gid, groups和SELinux security context
- 通过chcon命令修改文件的SELinux security context
- 通过restorecon还原文件默认的SELinux security context
- 通过runcon命令使程序运行在指定的SELinux security context

Android安全-----SEAndroid

/system/sepolicy/下

- service_contexts
- seapp_contexts
- file_contexts
- access_vectors
- global_macros
- te_macros
- genfs_contexts
- property_contexts
- ioctl_macros
- security_classes

Android安全-----SEAndroid

- SELinux 依靠标签来匹配操作和政策。标签用于决定允许的事项。套接字、文件和进程在 SELinux 中都有标签。SELinux 决定基本上是根据为这些对象分配的标签以及定义这些对象可以如何交互的政策做出的。
- 在 SELinux 中，标签采用以下形式：user:role:type:mls_level
- 其中 type 是访问决定的主要组成部分，可通过构成标签的其他组成部分进行修改。SEAndroid主要通过Label中的types来定义安全类型, 所以又被称为Type Enforment(这也是.te文件后缀的由来)

Android安全-----SEAndroid

user:role:type:mls_level

- user:　目前只有"u"这一种

  /system/sepolicy/users
- role:　可选值固定，主体---->r　客体----> object_r

  /system/sepolicy/roles
- type:　各种type类型, 比如device type, process type(domain), file system type, network type, IPC type等

  /system/sepolicy/domain.te, ...
- Security_level ：　s0 - mls_systemhigh

Android安全-----SEAndroid

TE文件政策规则采用以下形式：

allow domains types:classes permissions;

其中：

- Domain - 一个进程或一组进程的标签。也称为域类型，因为它只是指进程的类型。
- Type - 一个对象（例如，文件、套接字）或一组对象的标签。
- Class - 要访问的对象（例如，文件、套接字）的类型。
  /system/sepolicy/security_classes
- Permission - 要执行的操作（例如，读取、写入）。
  /system/sepolicy/access_vectors
  allow appdomain app_data_file:file rw_file_perms;

Android安全-----SEAndroid

不需要直接修改 system/sepolicy 中的文件而只需添加您自己的设备专用政策文件（位于/device/manufacturer/device-name/sepolicy 目录中）即可。

实现 SELinux，您必须创建或修改以下文件:

- 新的 SELinux 政策源代码 (*.te) 文件 - 位于 /device/manufacturer/device-name/sepolicy 目录中。这些文件用于定义域及其标签。在编译到单个 SELinux 内核政策文件时，新的政策文件会与现有的政策文件组合在一起。
- 更新后的 BoardConfig.mk makefile - 位于包含 sepolicy 子目录的目录中。如果初始实现中没有 sepolicy 子目录，那么在该子目录创建之，必须更新 BoardConfig.mk makefile，以引用该子目录。

Android安全-----SEAndroid

- file_contexts - 位于 sepolicy 子目录中。该文件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，您必须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。
- genfs_contexts - 位于 sepolicy 子目录中。该文件用于为不支持扩展属性的文件系统（例如，proc 或 vfat）分配标签。
- property_contexts - 位于 sepolicy 子目录中。该文件用于为 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。

Android安全-----SEAndroid

- service_contexts - 位于 sepolicy 子目录中。该文件用于为 Android Binder 服务分配标签，以便控制哪些进行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。
- seapp_contexts - 位于 sepolicy 子目录中。该文件用于为应用进程和 /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。
- mac_permissions.xml - 位于 sepolicy 子目录中。该文件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。然后，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。

Android安全-----SEAndroid

- 修改需要的文件后, 只需在 sepolicy 子目录和各个政策文件创建之后，更新 BoardConfig.mk Makefile（位于包含 sepolicy 子目录的目录中）以引用该子目录和这些政策文件即可.

```
BOARD_SEPOLICY_DIRS += \
        <root>/device/manufacturer/device-name/sepolicy
```

Android安全-----SEAndroid

- 1. 在内核和配置中添加 SELinux 支持。
- 2. 为通过 init 启动的每项服务（进程或守护进程）分配专用的域。
- 3. 通过以下方式标识这些服务：

  . 查看 init.<device>.rc 文件并找到所有服务。

  . 检查 dmesg 输出中以下形式的警告：“init: Warning! Service name needs a SELin ux domain defined; please fix!”

  （init：警告！服务名称需要一个已定义的 SELinux 域；请更正！）。

  . 检查 ps -Z | grep init 输出，看看哪些服务正在 init 域中运行。
- 4. 为所有新进程、驱动程序、套接字等添加标签。需要为所有对象添加适当的标签，以确保它们能够与您应用的政策正确交互。请参阅 AOSP 中使用的标签，以便在创建标签名称时参考。
- 5. 制定全面涵盖所有标签的安全政策，并将权限限定到其绝对最低级别

Android安全-----SEAndroid

当开始着手自定义 SELinux 时，制造商应记得做以下事情：

为所有新的守护进程编写 SELinux 政策

尽可能使用预定义的域

为作为 init 服务衍生的所有进程分配域

在编写政策之前先熟悉相关的宏

向 AOSP 提交对核心政策进行的更改

不要做以下事情：

- 创建不兼容的政策
- 允许对最终用户政策进行自定义
- 允许对 MDM 政策进行自定义
- 恐吓违反政策的用户
- 添加后门程序

Android安全-----SEAndroid

调试:

SELinux默认是enforing模式, 调试时会传命令给内核设置成permissive模式(androidboot.selinux=permissive), 在重新设置回enforing模式之前, 需要解决所有的avc denial.

--adb shell su 0 dmesg | grep avc

--adb logcat | grep avc

avc denial例子:

- avc: denied { connectto } for pid=2671 comm="ping" path="/dev/socket/dnsproxyd" scontext=u:r:shell:s0 tcontext=u:r:netd:s0 tclass=unix_stream_socket
- <5> type=1400 audit: avc: denied { read write } for pid=177
- comm="rmt_storage" name="mem" dev="tmpfs" ino=6004 scontext=u:r:rmt:s0
- tcontext=u:object_r:kmem_device:s0 tclass=chr_file

Android安全-----SEAndroid

- adb shell su 0 cat /proc/kmsg > dmesg.txt &
- adb logcat > logcat.txt &

使用 audit2allow

selinux/policycoreutils/audit2allow 工具可以获取 dmesg 拒绝事件并将其转换成相应的 SELinux 政策声明。因此，该工具有助于大幅加快 SELinux 开发速度。

- audit2allow -p out/target/product/device/root/sepolicy < dmesg.txt

展开

- 着重扩展SElinux(TEAC, MLS), 简要了解RBAC, Constraint
- 对比下SElinux和SEAndroid的细微差别
- 分析一些重要的sepolicy文件(file_contexts, access_vectors,...)
- 分析一些te文件和例子
- 配置简单的te文件(ep: 如何启动一个进程?)
- 文件(系统文件&App文件)/进程(daemon&app)安全上下文的设置
- SEAndroid的生成过程, 以及Android 7与Android 8的区别

- https://source.android.com/security/
- https://source.android.com/devices/tech/config/filesystem
- http://blog.csdn.net/luoshengyang/article/details/35392905
- http://www.cnblogs.com/shell812/p/6379321.html