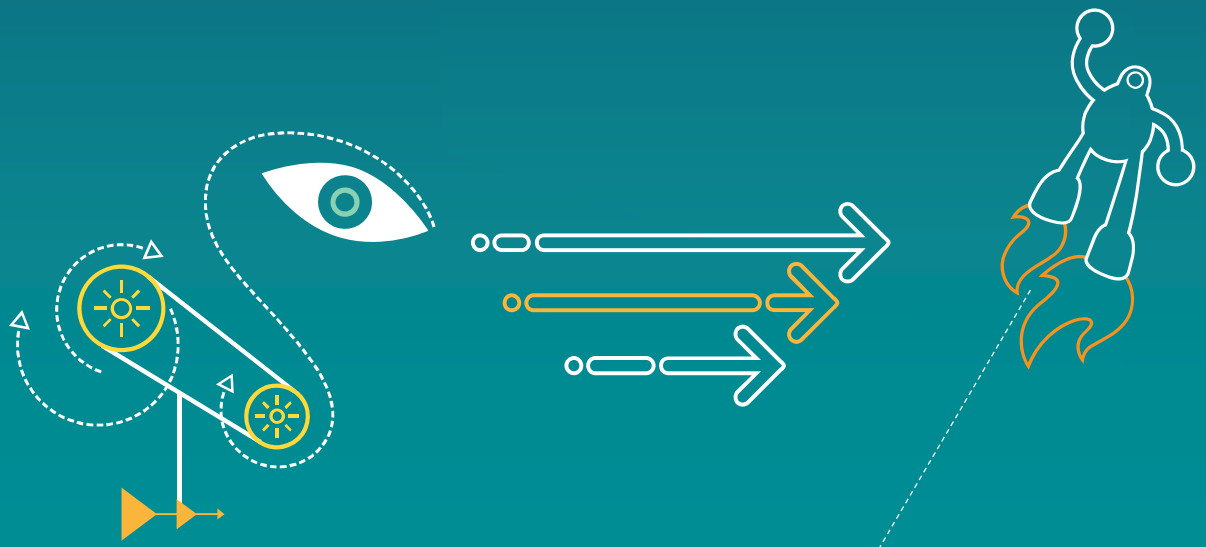
Ravi Kumar Siddojigari
16 Feb 2018

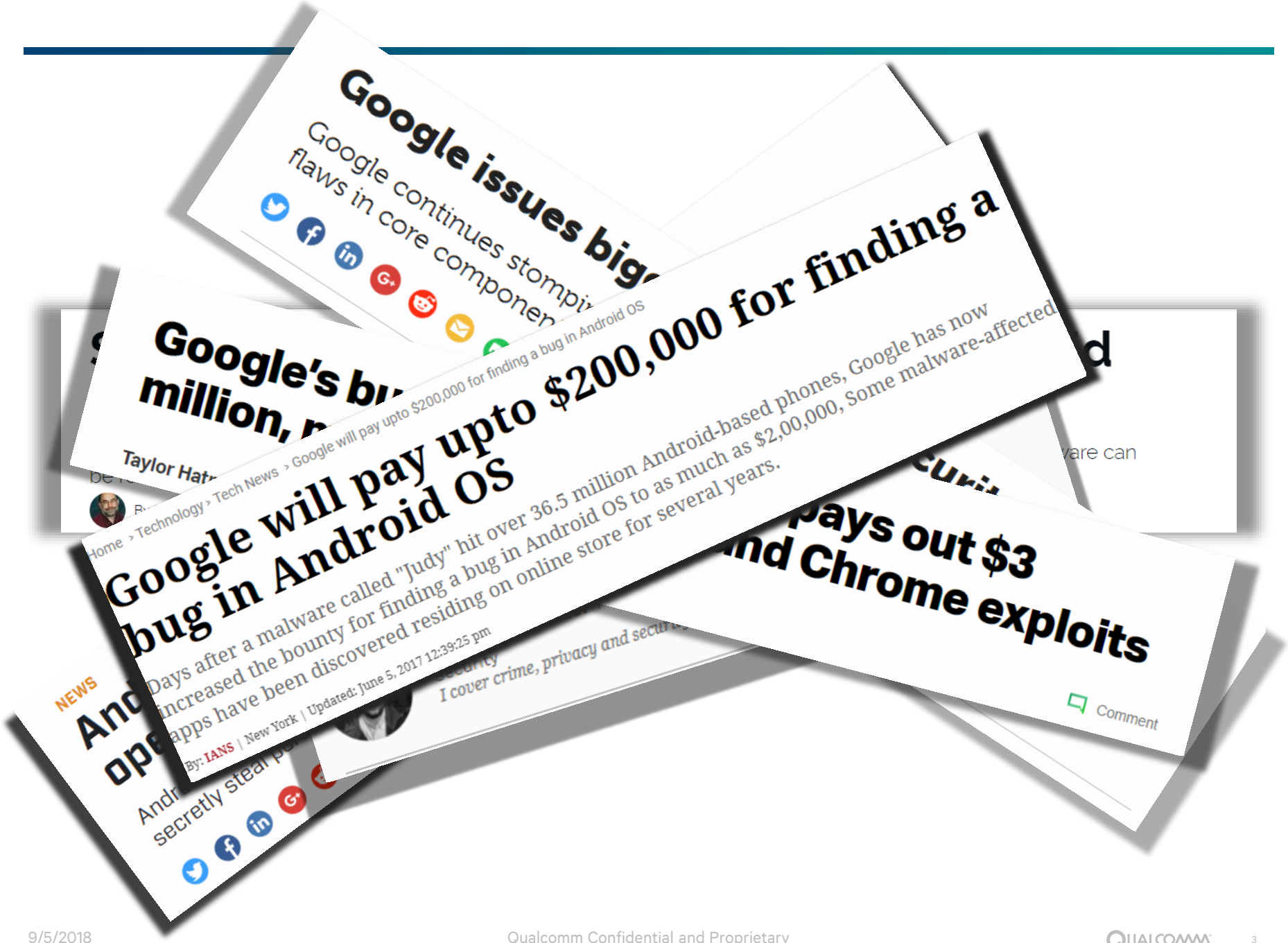
Overview
SELinux and Treble

QUALCOMM®



Agenda

- Overview of SELinux
- Basic datatype and Sepolicy
- Treble and SELinux
- General issue and Guideline for CE issue .


Google issues big
Google continues stompi
flaws in core componen
Google's bu
million,
Taylor Hatt
Google will pay upto $200,000 for finding a bug in Android OS
Home › Technology › Tech News › Google will pay upto $200,000 for finding a bug in Android OS
Google will pay upto $200,000 for finding a bug in Android OS
Days after a malware called "Judy" hit over 36.5 million Android-based phones, Google has now increased the bounty for finding a bug in Android OS to as much as $2,00,000, Some malware-affected apps have been discovered residing on online store for several years.
By: IANS | New York | Updated: June 5, 2017 12:39:25 pm
I cover crime, privacy and secu
pays out $3
nd Chrome exploits
Comment
NEWS
And
ope
secretly steal pe

To counter this security issues

- Starting patching policy （ Security bulletin –monthly patches ）

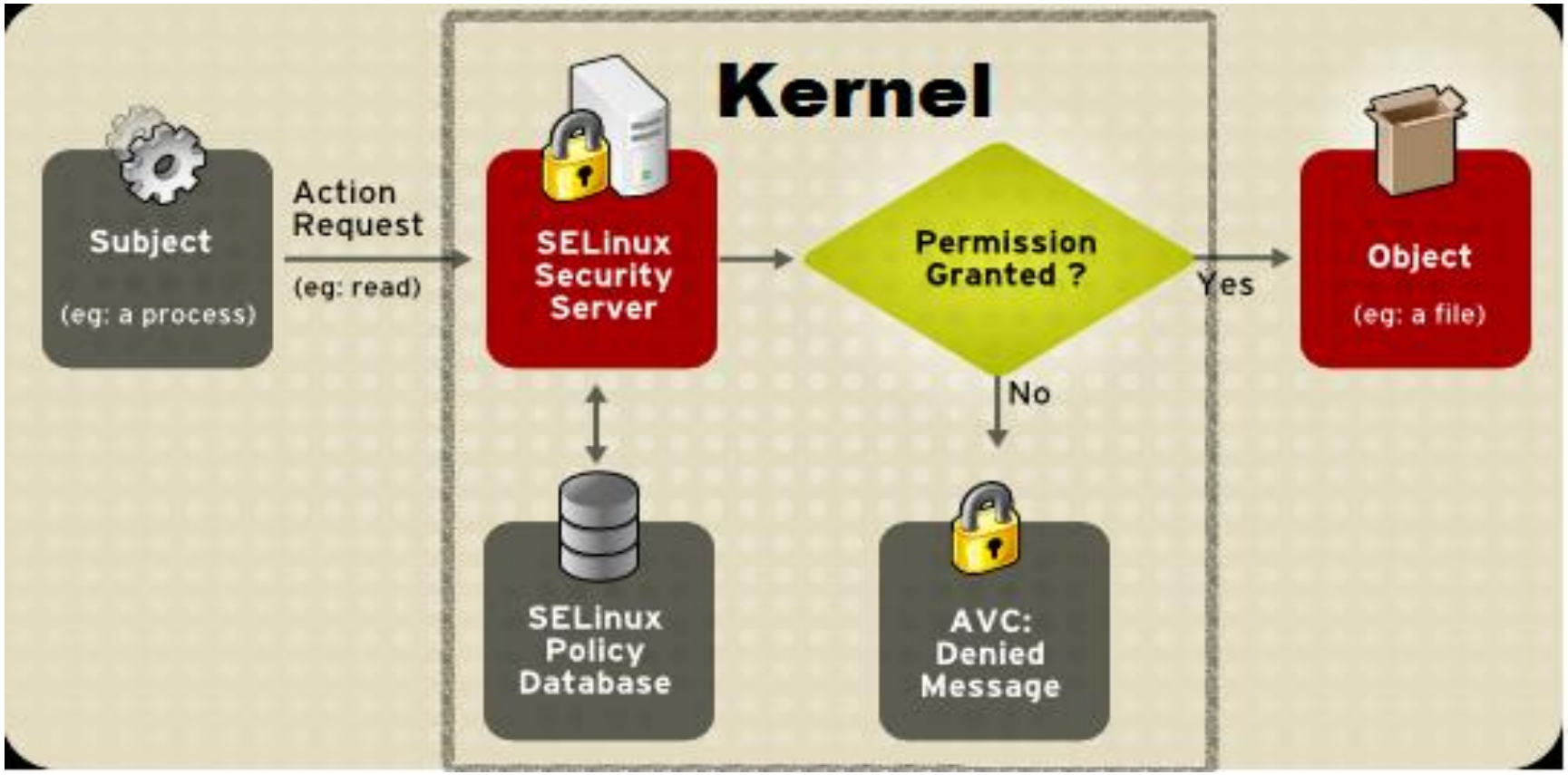
https://source.android.com/security/bulletin/

- Hardening security
  - SELinux
  - Scanning tools (ASAN/KASAN/Fuzzer ...)
  - Compiler tools and check enabled (like BIONIC FORTIFY)
  - Restriction of capabilities of services and groups.
  - Signed modules apk , CA.
- Upgrading the compliance tests suite
  - Security Compliance checks (CTS /STS)

In short what is SELinux

- Security-Enhanced Linux (SELinux) was invented by the National Security Agency, circa 2003
- Widely accepted Mandatory Access Control (MAC) model
- SELinux fundamentally labels all users, processes and files, and then grants these objects the minimum of permissions to accomplish their task. In the event that an object attempts to perform anything out of the scope of its security policies, it's stopped and then reported.
- The SELinux implementation uses role-based access control (RBAC), which provides abstracted user-level control based on roles, and Type Enforcement® (TE).
- TE uses a table, or matrix to handle access controls, enforcing policy rules based on the types of processes and objects.
- Process types are called domains

Simple control flow in terms of SElinux



- Subject : is mostly the process that is request for some operation on resource .
- SS : security Server is Linux driver which is going to handle sys_call that are hooked to LSM
- Policy Database : is Pre-defined permission set which are defined by the during build process .
- Object : Resource that is been access by the subject (can be file / device node /socket/another services …)

Advantage of SELinux

- Wider / fine grain control of permission against legacy DAC where only read/write /execute can be controlled . Here on file
  getattr, relabelto, unlink ,ioctl, execute, append,read,setattr,swapon,write,lock,create,rename,link ...

https://selinuxproject.org/page/ObjectClassesPerms

- Defines and enforces a system-wide security policy
- Over all processes, objects, and operations.
- Can confine flawed and malicious applications. Even ones that run as “root” / uid 0.
- Can prevent privilege escalation.

SELinux-Operation modes

- Permissive mode :
  - Security server will not do a denials of service (DoS) but just logs the avc denials giving the hint on missing permission .
  - Used in development environment /debug environment where we are not sure of access required .
  - Can be set via Kernel cmdline " androidboot.selinux=permissive "
  - Can be set via adb shell " adb shell setenforce 0" non-persist across reboot .

  Setenforce make selinux to permissive starting from that point (not from bootup)
- Enforce mode :
  - Strictly go by the documented sepolicy and will do a denials (logs the denials messages)
  - End product is expected to be in enforce mode .
  - Default bootup mode is always enforced .
- Disable :
  - SELinux disable mode is not supported in Android (from N) disabling this will show up bootup issue (init.cpp always assumes selinux is enabled) .

Label and its parts

Labels are defined in following format

user:role:type[:range]

Where:

`user`	The SELinux user identity. This can be associated to one or more roles that the SELinux user is allowed to use.
`role`	The SELinux role. This can be associated to one or more types the SELinux user is allowed to access.
`type`	When a type is associated with a process, it defines what processes (or domains) the SELinux user (the subject) can access. When a type is associated with an object, it defines what access permissions the SELinux user has to that object.
`range`	This field can also be know as a `level` and is only present if the policy supports MCS or MLS. The entry can consist of: ▪ A single security `level` that contains a sensitivity level and zero or more categories (e.g. `s0`, `s1:c0`, `s7:c10.c15`). ▪ A `range` that consists of two security levels (a low and high) separated by a hyphen (e.g. `s0 - s15:c0.c1023`). These components are discussed in the Security Levels section.

Examples :

/dev/socket/rild3 u:object_r:rild_socket:s0

/vendor/bin/rmt_storage u:object_r:rmt_storage_exec:s0

/vendor/bin/hw/android\.hardware\.keymaster@3\.0-service-qti u:object_r:hal_keymaster_qti_exec:s0

In Android user is always “u” , Role is always “Object_r “ and range is always “S0”

Checking the mode and labels

- getenforce and setenforce are 2 utilities which can be used .

```
[root@localhost ~]# getenforce
Enforcing
```

- Other utilities
  - ls -Zl will list the file/dir labels
  - ps –ZA will list the process labels
  - getprop –Z will list the property labels

Setenforce work on root shell and will work on debug /Engg build . End product should not allow this .

In Android , if we are not defining labels for file/folder it will inherit from the label from parent folder .

Any property that is not labeled will get

u:object_r:default_prop:s0 ---- for system property

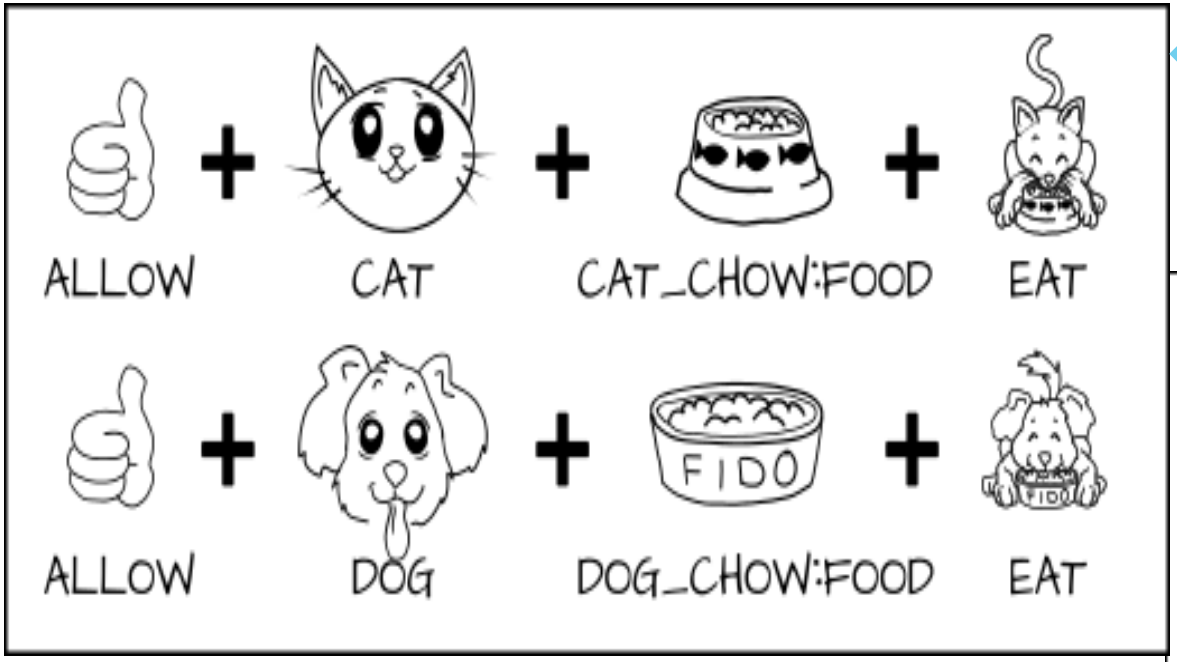
u:object_r:vendor_default_prop:s0 -- for vendor property

Files that are used for labeling …

- file_contexts – used for file labeling
- genfs_contexts – used for file labeling which are created @ runtime
- hwservice_contexts– used for labeling hwservice
- property_contexts -- used for labeling property
- seapp_contexts --- used for creating domain for apk or java serives
- service_contexts – used for labeling the serives (System side)
- vndservice_contexts -- used for labeling the vendor services

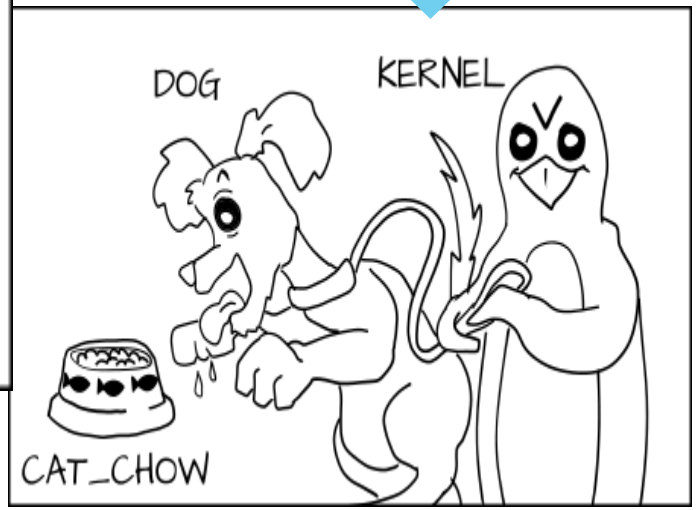
Typical Example

- Cat and Dog are 2 different process
- Cat_chow and Dog_chow is object
- Food is the class
- Eat is type of verb /action



Allow example

Neverallow example



Ref: https://opensource.com/business/13/11/selinux-policy-guide

TE file example

Permission needed by the service are document in te files and using same name as services will be easy to read as an example for audiod te file for audiod process below

xref: /[redacted]/device/qcom/sepolicy/vendor/common/audiod.te

Home | History | Annotate | Line# | Navigate | Raw | Download

```
1  # audio daemon # is for comments
2  type audiod, domain;   saying audiod is a domain
3  type audiod_exec, exec_type, vendor_file_type, file_type
4  init_daemon_domain(audiod)
5  allow audiod proc_audiod:file r_file_perms;
6  allow audiod audio_device:chr_file rw_file_perms;
7  #allow audiod audioserver_service:service_manager find;
8  #binder_use(audiod)
9  binder_call(audiod, audioserver)
10
```

is of vendor_file_type and exec_type if you see the file_context for audio_exec its defined as "/vendor/bin/audiod u:object_r:audiod_exec:s0"

start of audio is from init script

permission for audiod to read proc_audiod file.
where proc_audiod is again a label to /proc/asound/card0/state as defined in genfs_context as below
genfscon proc /asound/card0/state u:object_r:proc_audiod:s0

adding binder call permission this is a macro defination can be seen in te_macro/global_macro

Code tree .

- SELinux rules are document in the following project
  - device/qcom/sepolicy（owned by Qcom）
  - system/sepolicy（part of AOSP project）

Overall Stack security policy（global security rules）are controlled by Google via , Never allows , Restrictions in sepolicy rule in AOSP project (system/sepolicy).

Google added test cases to check to make sure no one tamper this global security rules (as part of CTS /VTS /STS) .

Customers are advised not to add /edit the system/sepolicy project

Build Variable used in Sepolicy

.BOARD_PUBLIC_PLAT_SEPOLICY_DIRS:

- Contains only the type definition : create a new te file for the daemon and add the type, ex: type dpmd, domain, coredomain; -> coredomain is a must for domain in system partition

BOARD_PUBLIC_PRIV_SEPOLICY_DIRS:

- Type Definition in this folder are not accessible to vendor and it only contains policy for daemons that resides in system partition. This requires both source and target context to reside on system partition, if both resides in system, then labels for those has to be in this folder.
- If the label is going to be used by vendor policy, then define the label in BOARD_PUBLIC_PLAT_SEPOLICY_DIRS folder.

BOARD_SEPOLICY_DIRS

- All device/vendor specific policy goes here.

Assumption and References:

- BOARD_PUBLIC_PLAT_SEPOLICY_DIRS: device/qcom/sepolicy/public
- BOARD_PUBLIC_PRIV_SEPOLICY_DIRS: device/qcom/sepolicy/private
- BOARD_SEPOLICY_DIRS: device/qcom/sepolicy/vendor/common, test and other directories.

Customer are advised to create there own folder parallel to /device/qcom as device/<customer_product>/sepolicy and use the above variables .

General guidelines

- Every services should run in its own domain.
- Transition of domain is not allowed .
- Domain name can be shared

Example : if we don't define a domain it will fail to start .

[31.160512] init: Could not start service 'crashdata-sh' as part of class 'late_start': File /vendor/bin/init.qcom.crashdata.sh(labeled "u:object_r:vendor_file:s0") has incorrect label or no domain transition from u:r:init:s0 to another SELinux domain defined. Have you configured your service correctly? https://source.android.com/security/selinux/device-policy#label_new_services_and_address_denials

In the above case init.qcom.crashdata.sh is missing selinux label and landing to a default label called vendor_file(all file on /vendor will get this label if not defined)

Understanding the AVC denial

- [16.318752] type=1400 audit(529.779:162): avc: denied { write } for pid=469 comm="b2g" name="alarm" dev="tmpfs" ino=1158 scontext=u:r:init:s0 tcontext=u:object_r:alarm_device:s0 tclass=chr_file permissive=1

{....}	operation that it was trying to do
Scontext	Source label that was trying to do this operation
Tcontext	resource label on which this action / permission is missing
Tclass	Type of resource . You can check different class and operation support on such resources @ http://selinuxproject.org/page/NB_ObjectClassesPermissions
Permissive	is it true / false ?
Ino	Node on which it was trying (fs numbering)

Understanding AVC denial …

Example denial message:

E/SELinux (608): avc: denied { add } for service=seempservice pid=1456 uid=1000

scontext=u:r:seempd:s0 tcontext=u:object_r:default_android_service:s0 tclass=service_manager

tclass may be service_manager, vndservice_manager, hwservice_manager. Based on tclass define the service in the respective file.

service_contexts

seempservice u:object_r:seemp_service:s0

service.te

type seemp_service, service_manager_type;

vndservice_contexts

vendor.qcom.PeripheralManager u:object_r:per_mgr_service:s0

vndservice.te

type per_mgr_service, vndservice_manager_type;

hwservice_contexts

vendor.qti.hardware.radio.atcmdfwd::IAtCmdFwd u:object_r:hal_atfwd_hwservice:s0

hwservice.te

type hal_atfwd_hwservice, hwservice_manager_type;

Domains

Coredomains

- works from system partition
- Access is restricted to system resources
- access to /data/vendor is not allowed.

Vendor_init domain

- vendor *.rc files run in this domain .

Vendor_shell domain

- Vendor process when using shell utility will invoke vendor_shell

Untrusted_app/system_app/platform_app

- most of the 3rd party apps /apk which are (not signed)run as untrusted domain , based on singed process takes system_app/ platform_app domain.

Qti_init_Shell domain

used for qualcomm init related shell scripts .

Example for HAL

type mediaserver, domain, coredomain; -> **coredomain** only if its defined in private folder and is part of system image

If you want the domain definition to be accessible to vendor, define only type mediaserver, domain in public folder.

type mediaserver_exec, exec_type, vendor_file_type[for android O], file_type; -> **vendor_file_type** if binary resides in vendor partition

Define the HAL binary label in file_contexts:
(vendor|system/vendor)/bin/hw/android\.hardware\.sensors@1\.0-service u:object_r:hal_sensors_qti_exec:s0

For backward compatibility add the system/vendor/bin as well

Host the Hal using below server domain macro.
hal_server_domain(hal_sensors_qti, hal_sensors)

Define the attribute for client/server and the hal
hal_sensor, hal_sensor_client, hal_sensor_server

For client to communicate with server in client.te file
hal_client_domain(clientdomain, hal_sensor) , ex: hal_client_domain(system_server, hal_sensor)

Allow all client to communicate with server
binder_call(hal_sensors_client, hal_sensor_server). DO NOT USE binder_call(system_server, hal_sensor)
binder_call(hal_sensors_server, hal_sensor_client).

Note: if you are not hosting server, then use audit2allow to generate rule and replace appropriate one with **macro(te_macros)**. hal_server_domain should not be used for HAL which require communication with network socket or net_raw/net_admin capabilities,

Are we on track ?

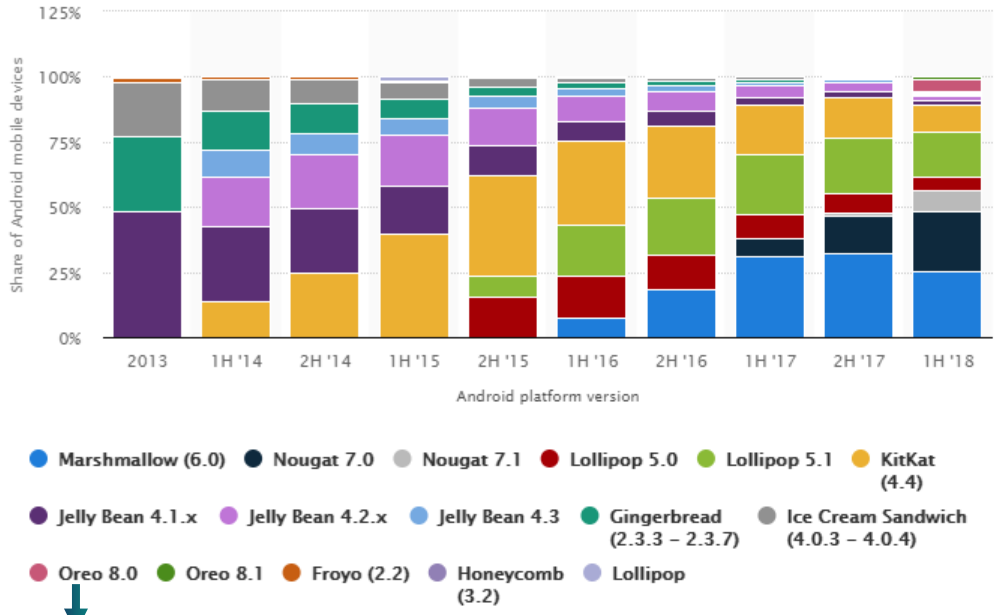
statista — The Statistics Portal
Statistics and Studies from more than 22,500 Sources

Enter search term, e.g. social media

Prices & Access | Statistics | Reports | Expert Tools | Infographics | Services | Global Survey NEW | Login

Technology & Telecommunications › Software › Android version market share 2018

Android version market share distribution among smartphone owners as of in February 2018



DOWNLOAD | SETTINGS | SHARE

PNG | PDF | XLS | PPT

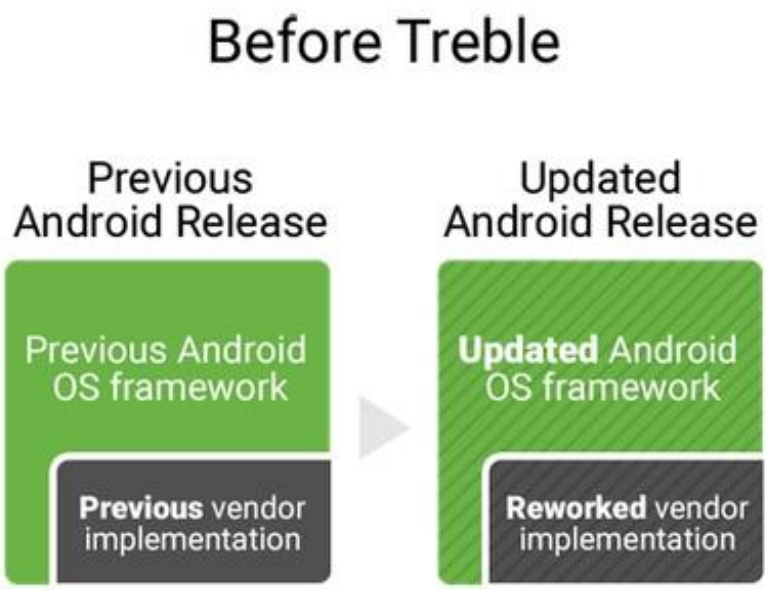
DESCRIPTION | SOURCE | MORE INFORMATION

In this graphic you can see the Android version market share distribution among smartphone owners in February 2018. This month, Android version 5.1 (Lollipop) had a market share of 19.2 percent among smartphone owners featuring Google's Android operating system. The figures are based on the number of Android devices that have accessed the Google Play Store within a 7-day period ending on February 5th, 2018.

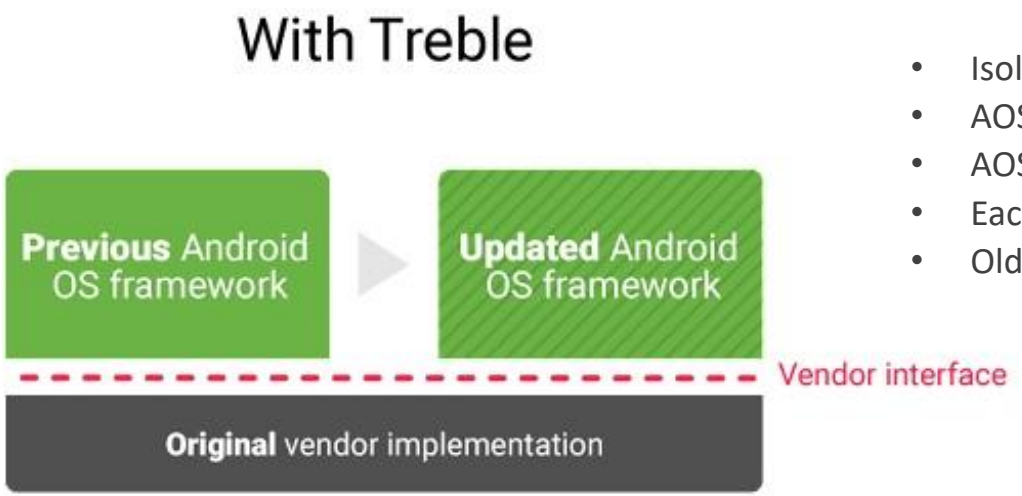
Google Android versions - additional information

Since the first quarter of 2011, Google's mobile

TREBLE



- Vendor mixed his code to AOSP
- Each upgrade of AOSP need again a rework
- Issues not common across the devices AOSP may not be common.
- Each case TA cycle is high
- Mix of older vendor code + newer AOSP not possible or risker



- Isolated AOSP
- AOSP can be upgrade independently (GSI images)
- AOSP issues common across the devices
- Each case TA cycle is low
- Older vendor + newer AOSP is possible.

Treble's goal of independent updates for platform and vendor components means that ownership must be clearly defined for each object.

Type/attribute name spacing

- type foo, domain; ⟶ type vendor_foo, domain;

System Property and process labeling ownership

- foo.xxx ⟶ vendor.foo.xxx
- ro.foo.xxx ⟶ ro.vendor.foo.xxx
- persist.foo.xxx ⟶ persist.vendor.foo.xxx

Procfs (/proc)

- Only platform policy labels /proc . If vendor processes need access to files in /proc that are currently labeled with the default label (proc), vendor policy MUST not explicitly label them and should instead use the generic proc type to add rules for vendor domains. This allows the platform updates to accommodate future kernelinterfaces exposed through procfs and label them explicitly as needed.

Debugfs (/sys/kernel/debug)

- In the short term, only vendor policy may label debugfs . In the long term, remove debugfs.

Tracefs (/sys/kernel/debug/tracing)

- Only platform may label tracefs.

Object ownership and labeling

Sysfs (/sys)

- The platform may label only the select files listed Otherwise, only vendor may label files.

tmpfs (/dev)

- Vendor may label only files in /dev/vendor (e.g., /dev/vendor/foo , /dev/vendor/socket/bar).

Rootfs (/)

- Only system may label files in / .

Data (/data)

- Disallow vendor labeling outside /data/vendor . Only platform may label other parts of /data.

Continuance .....

- On TREBLE devices, vendor and system components are only allowed to share files by passing open FDs over hwbinder.
- Ban all directory access and all file accesses other than what can be applied to an open FD such as ioctl/stat/read/write/append. This is enforced by segregating /data.
- Vendor domains may directly access file in /data/vendor by path, but may only access files outside of /data/vendor via an open FD passed over hwbinder.
- Likewise, core domains may only directly access files outside /data/vendor by
- path and files in /data/vendor by open FD.

- Dir creation /file creation / ownership change
  - Should be done in rc files .
  - Group of that file/folder need to be subscribed for the respective service /process which need access.
  - Any folder that is created and labeled on /data should have data_file_type attribute .

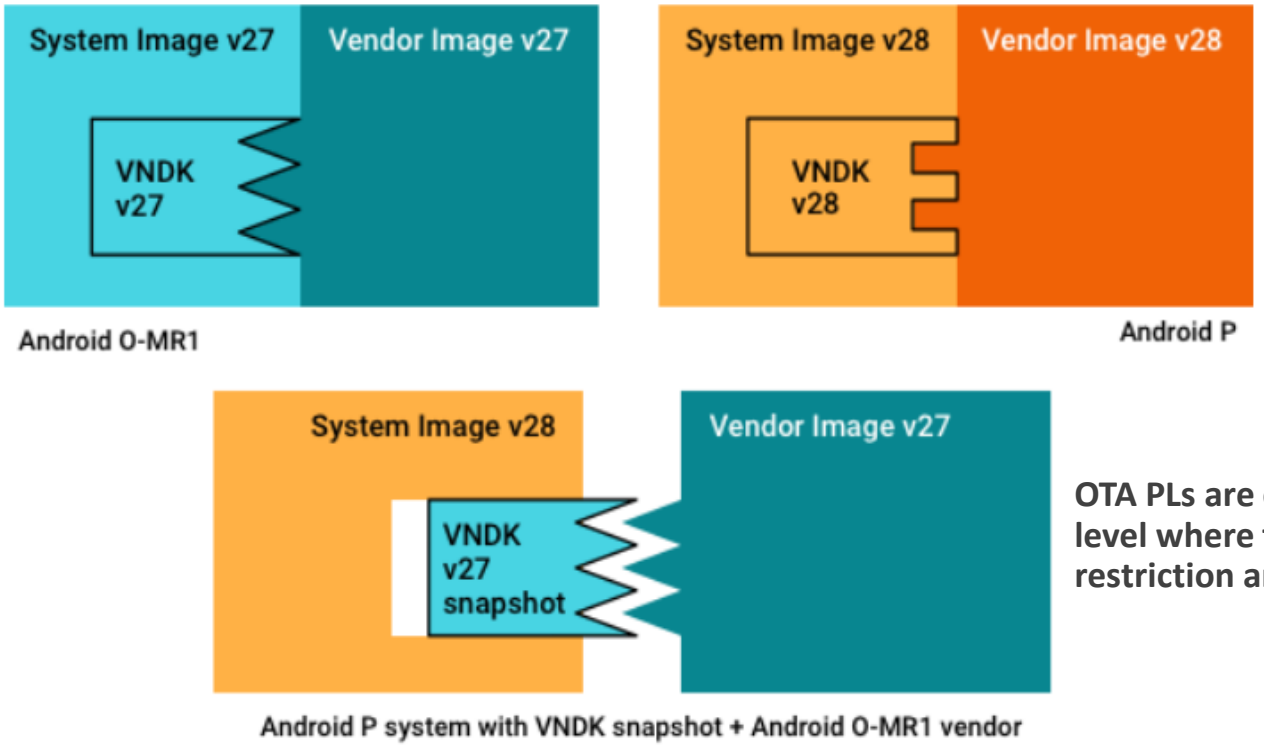
Continuance….

- Usage of dac_override and dac_read_search
  - Instead of getting these permissions, it is better to add the process to a group or change the permissions of the files it tries to access
- Disallow coredomains from accessing vendor_files on Treble
- Coredomains cannot execute vendor code
- Coredomains should not set vendor property
- Vendordomains should not set Coredomain property (exception to whitelisted property recommend to check http://opengrok.qualcomm.com/source/xref/LA.UM.7.3/system/sepolicy/public/property_contexts)

Typical upgrade on treble

Example: Upgrading system image only

Must include the VNDK snapshot and linker namespace configuration files for the vendor image in the system image. The linker namespace configuration files are automatically configured to search for VNDK libraries in `/system/lib[64]/vndk-${VER}` and `/system/lib[64]/vndk-sp-${VER}`.



OTA PLs are expected to define first shipping API level where from Selinux point of view the newer restriction are given exception .

Figure 1. Upgrading system only

Debugging guide

- ls –Zl <folder/filename> -- will show the label been set
- ps –ZA|grep <process name> -- will show the process running in which domain.
- dmesgl grep avc --- will show the current denials
- logcat | grep avc --- will show the current denials
- audit2allow -i <log with avc denails> -- to get supporting rule

  run audit2allow in new shell and may need policycoreutils package .
- getprop –Z | grep <property_name> - to get the label set for property
- service list or vndservice list - check if service is not able to find and seeing avc denials for services as Vendor service list and system service are different across access is not possible.
- Cat /vendor/etc/selinux/plat_sepolicy_vers.txt – sepolicy version
- Context files on target can be check in /etc/selinux/*_context

Debugging guide...

- Decoding the node
  - Example :avc: denied { read } for comm="time_daemon" name="name" dev="sysfs" ino=33906 scontext=u:r:time_daemon:s0 tcontext=u:object_r:sysfs:s0 tclass=file permissive=0

  #find /sys -inum 33906 will give the path that is been tried for access.
- Permissive logs can get more info then enforced logs so better to always ask customer issue with permissive (in permissive it should pass) .
- Most of time customer issue might be already fixed in internal tree so just try to check if the permission are adding in the local tree based on AVC denials /audit2allow output
- Any denials that are seen by customer customization cant be mainline .
- SDcard usage from vendor process is restricted and appdomains/system process can access this .

Debugging guide....

- No rules are allowed for default_props / default_service /vendor_default_props (restricted by neverallow) .
- Checking the use case in permissive and if works may not be an selinux issue it could be even TREBLE compliance issue .
- Untrusted_apps are 3rd party app where we don't have control , they can ask permission which may not be in security equation and we don't add such permission .
- Audit2allow may not always give a rule which fits the security equation .
- For debugging startup issue use serial log/dmesg logs and for other userspace issues use logcat logs.
- SELinux team may not be aware on what all permission are needed by a driver and tech team are the Right PoC to tell us why they need it.

Diag usage

- Diag is a diagnostic node that need to be used by developers and or production house.
- Exposing this in end-userbuild may lead to security risk so Qualcomm doesn’t mainline allowing this in user builds .
- Customer can do there security assessment and if needed they can allow this (as Qualcomm is not sure on how its used in customer code)

Quick links/ contacts .

- Qwiki's
  - Go/selinux
  - Go/seandroid
  - Go/SEAndroidworkflow
- Email groups:
  - android.sepolicy.approvers -- for sepolicy change approvers
  - Seandroid.support --- for queries and support related.
- PoC list :
  - Sridhar Parasuram /Paul Biswajit (SD)
  - Ravi Kumar Siddojigari /Jaihind Yadav (QIPL)
  - Adinarayana Gupta Grandhi (Lead)

Additional resources

- https://selinuxproject.org/
- https://developer.android.com/about/versions/pie/
- https://source.android.com/setup/build/gsi
- https://source.android.com/devices/architecture/vndk/linker-namespace
- https://source.android.com/setup/build/gsi#changes-to-keymaster-behavior
- https://source.android.com/devices/architecture/vndk/snapshot-design

Use for Qualcomm Technologies, Inc.

Thank you

References in this presentation to "Qualcomm" may mean Qualcomm Incorporated, Qualcomm Technologies, Inc., and/or other subsidiaries or business units within the Qualcomm corporate structure, as applicable.

Qualcomm Incorporated includes Qualcomm's licensing business, QTL, and the vast majority of its patent portfolio. Qualcomm Technologies, Inc., a wholly-owned subsidiary of Qualcomm Incorporated, operates, along with its subsidiaries, substantially all of Qualcomm's engineering, research and development functions, and substantially all of its product and services businesses,
including its semiconductor business.

QUALCOMM