SELinux(K2C)

THIS PAGE IS NO LONGER MANTIANED AND TRACKED — PLEASE REFER : SELinux guide for QCLINUX

- 1. **Guidance To Tech Teams**
  - 1.1. **0.0 Guidance:**
    - **QC Upstream services need to up-stream respective policies**
    - **All QC Upstream services need to work with enforcing**
    - **Address all the Upstream service denials and upload respective patches in refpolicy**
    - **Guidance on the process will be shared soon (TBD) - AI on security**
    - **Any exceptions must be reviewed and approved by security leads**
  - 1.2. **1.0 Guidance:**
    - **All QC Upstream/downstream services need to work with enforcing**
    - **QC Upstream services need to maintain delta policies in refpolicy as patches**
    - **Downstream services maintain policies in meta-qti-bsp**
    - **Address all the Upstream service denials**
    - **Guidance on the process will be shared soon (TBD) - AI on security**
    - **No exceptions to run in permissive**
    - **Create a requirement CR to keep a service in permissive mode. We will be enforcing by 15th dec 2023**

2. Development plan

Sep '23	Oct '23	Nov '23
• Develop and merge required changes to get the adb shell with permissive mode.	• Develop and merge required changes to get the adb shell with enforcing mode. • Initially MONOLITHIC design will be chosen later will be transitioned to MODULAR design since in upstream by default supports MODULAR design was used.	• Develop and merge required changes for the transition from MONOLITHIC to MODULAR. • Rollback mechanism from MODULAR to MONOLITHIC.

3. Integration

- Will be maintaining our downstream policies of downstream services in "meta-qti-bsp" as a dynamic layer.
- While building the policies we will pick upstream policies (refpolicy + meta-qti-distro/patches), down stream policies for 1.0 build.
- Enabling Selinux by adding "selinux" to DISTRO_FEATURES and the package group "packagegroup-core-selinux" to CORE_IMAGE_BASE_INSTALL
- And we have chosen for multi level security by adding "refpolicy-mls" to PREFERRED_PROVIDER_virtual/refpolicy .
- Selinux Kernel configs were enabled using yocto ".cfg" format.
- There are two approaches to load policies:
  - Monolithic
    - All policies are compiled and grouped together in a single policy bin file "policy.<policy-version>".
    - We are using Kirkstone(yocto-version) for 1.0 and policy-version is "33".
  - Modular
    - A separate bin file with extension ".pp" will be generated for each service module.
    - And we load the policy file of a service just before the service runs
  - Both upstream refpolicy and meta-selinux were configured to "Modular" by default.
  - To choose Monolithic approach we need to set POLICY_MONOLITHIC to 'y'. Default value is 'n'.
- We will be maintaining sepolicies source tree similar to upstream refpolicy and copy our policy files to refpolicy workspace while building. So that the refpolicy will pick QC policies for compilation.
- **Gerrits:**
  - Selinux dynamic layer and kernel config fragment: https://review-android.quicinc.com/c/meta-qti-bsp/+/4865281 (Merged)
  - Recipe to build QC policies, build issue fixes: https://review-android.quicinc.com/c/meta-qti-bsp/+/4879258 (Merged)
  - Selinux addition to distro and package group addition to image recipe: https://review-android.quicinc.com/c/meta-qti-distro/+/4879260 (Merged)
  - Gerrit with miss kernel configs for selinux: QCLINUX: Add missing kernel configs for selinux (I1d9e702e) · Gerrit Code Review (quicinc.com) (Review under progress)
  - Distro change to pass file_context file to rootfs: PENDING: pass file_contexts file to mkfs.ext4 (I25232a76) · Gerrit Code Review (quicinc.com) (Review under progress)
  - e2fsprogs patch: PENDING: add feature to apply selinux labels to mkfs.ext4 (Idb4800a3) · Gerrit Code Review (quicinc.com) (Review under progress)

4. Current status

4.1. Phase-1: Get the adb shell with permissive mode following monolithic approach (InProgress)

- Dynamic layer and recipes are ready, yet to be merged.
  - Recipe is more generic so that external developer can simply pick our recipe to build their policies.
- Analyzed upstream ref-policy build process and able to build polices with monolithic approach.

4.1.1. Challenges:

- Build issues faced with upstream recipes in case of Monolithic approach – Fixed
- Path Mismatch issues for setfiles while generating builds – Fixed with necessary changes in recipes
- Refpolicy recipes doesn't separate monolithic and modular policy files – Fixed in meta-selinux (need to upstream the fix)

4.2. Phase-2: Get the adb shell with enforcing mode following monolithic approach (InProgress)

- Identify work around policies for device boot up in enforcing mode. (InProgress)

4.2.1. Challenges:

- Policies needs to be re-written as per upstream community guidelines. (Major work)
- Why not modular directly?
  - QC following monolithic currently in LE
  - No effective way to load upstream service modules at run time. – Need to debug
- Systemd bins are not getting labelled at run time even after labelling in file_contexts (Debug InProgress).

4.3. Phase-3: Get the adb shell with enforcing mode with modular approach (To be started)

- Verify dynamic loading of policies at runtime
- Need to work with upstream team consume the policies of services up streamed by QC.
- Need to Identify a solution if upstream doesn't accept policy changes for QC upstream services
- Roll back mechanism from modular to monolithic. (optional – only for documentation purpose for customers)

4.3.1. Challenges:

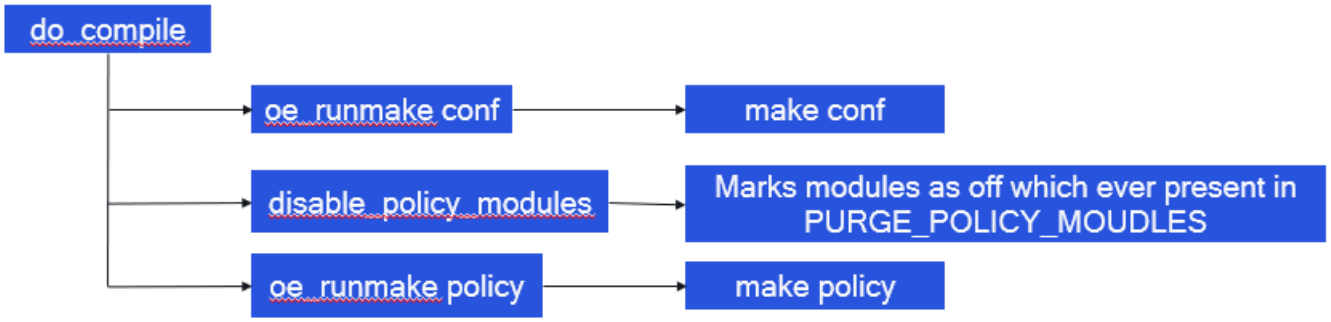
- If Selinux is disabled – Evaluate the impact on services where policy dependency set
- Not yet evaluated dependencies on loading policy modules for general upstream services both Selinux enabled/disabled

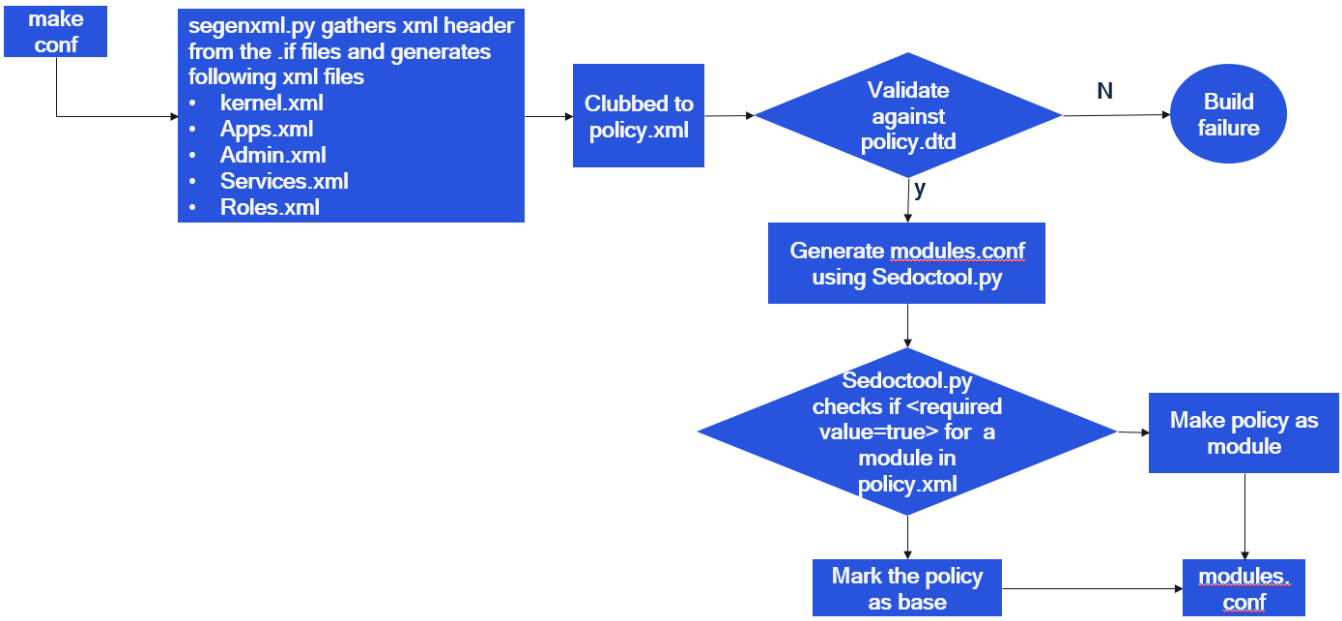
5. Why do Selinux need acl, pam xattr?

1. **ACL (Access Control Lists):** SELinux uses ACLs to provide fine-grained access control on files and directories. ACLs allow administrators to specify access permissions for users and groups beyond the traditional Unix file permissions (read, write, execute). This is useful for SELinux to define and enforce security policies based on user roles and specific permissions required by various processes.
2. **PAM (Pluggable Authentication Modules):** SELinux can integrate with PAM to enforce access controls based on authentication methods. PAM allows you to configure how users authenticate and gain access to system resources. SELinux can use PAM to control which users or processes are allowed to perform certain actions based on their authentication status.
3. **xattr (Extended Attributes):** Extended attributes in Linux are metadata attached to files and directories. SELinux uses extended attributes to store security context information that specifies how a file or process should be treated in terms of access control. xattr support is essential for SELinux to manage and enforce these security contexts.

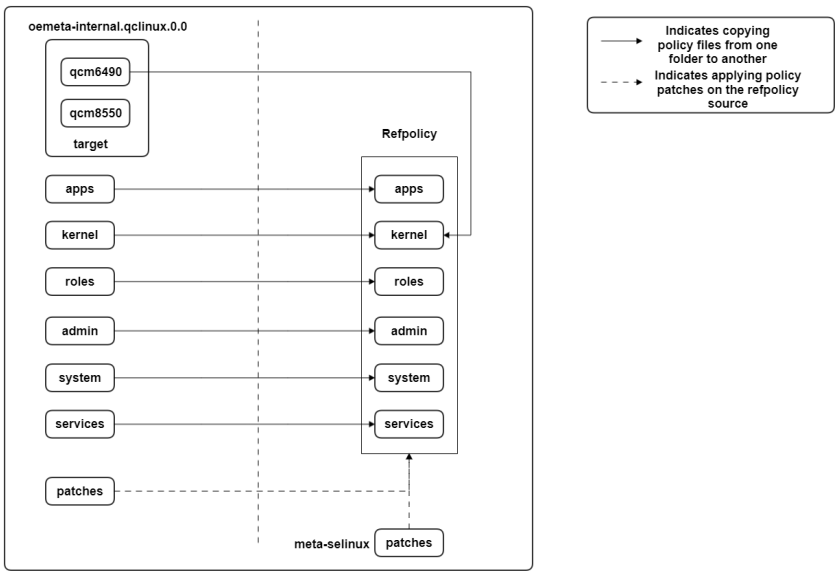
By including these components in DISTRO_FEATURES, the Linux distribution ensures that SELinux can fully leverage these features to enforce security policies, manage access control, and provide a more robust security framework. Together, these components help SELinux enforce Mandatory Access Control (MAC) policies that go beyond traditional discretionary access control mechanisms and enhance the overall security of the system.

6. Build flow (In progress)





7. **Fetch and Patch flow**


oemeta-internal.qclinux.0.0
qcm6490
qcm8550
target
Refpolicy
apps
kernel
roles
admin
system
services
patches
meta-selinux
patches
Indicates copying policy files from one folder to another
Indicates applying policy patches on the refpolicy source