

Overlays And Containers

Miklos Szeredi, Red Hat

Vivek Goyal, Red Hat

Introduction to overlays

Union or...?

- Union: all layers made equal
- How do you take the union of two files?
- Or a file and a directory?
- **NO!** Layers can't be treated equal

...overlay!

- Layer upon layer upon layer...
- Only upper layer can be modified
 - copy-up (exception: directory contents)
- Objects in one layer cover up objects with the same name in layer(s) below
- Exception: directories, which are merged
- Exception for the exception: opaque directories
- One more exception: whiteout
 - covers up anything and makes it look like nothing

Design

- **Userspace API (most important!)**
 - No new object types
 - Whiteout -> char dev with 0/0 device number
 - Opaque dir -> xattr
- **Make it as simple as possible (and not a bit simpler)**
 - Most of the logic is in a separate filesystem module
 - Some VFS impact but not much; some FS impact but not much
- **Upstream early**
 - It doesn't have to do *everything* right; features can be added later...

Implementation

- **Separate cache for the overlay directory tree**
 - Allows less impact on VFS/FS
 - **BUT** bad for memory use
- **Shared cache for the file contents**
 - Copy-up when opened for write (may be too early)
 - Ugliness when copy-up happens while file is already open read-only
 - **BUT** great for performance and memory use
- **Limitations**
 - modifying lower layer -> don't care
 - Not (yet) a "POSIX" filesystem (st_dev/ino quirks, directory rename, hard link copy-up, etc)

Features added later

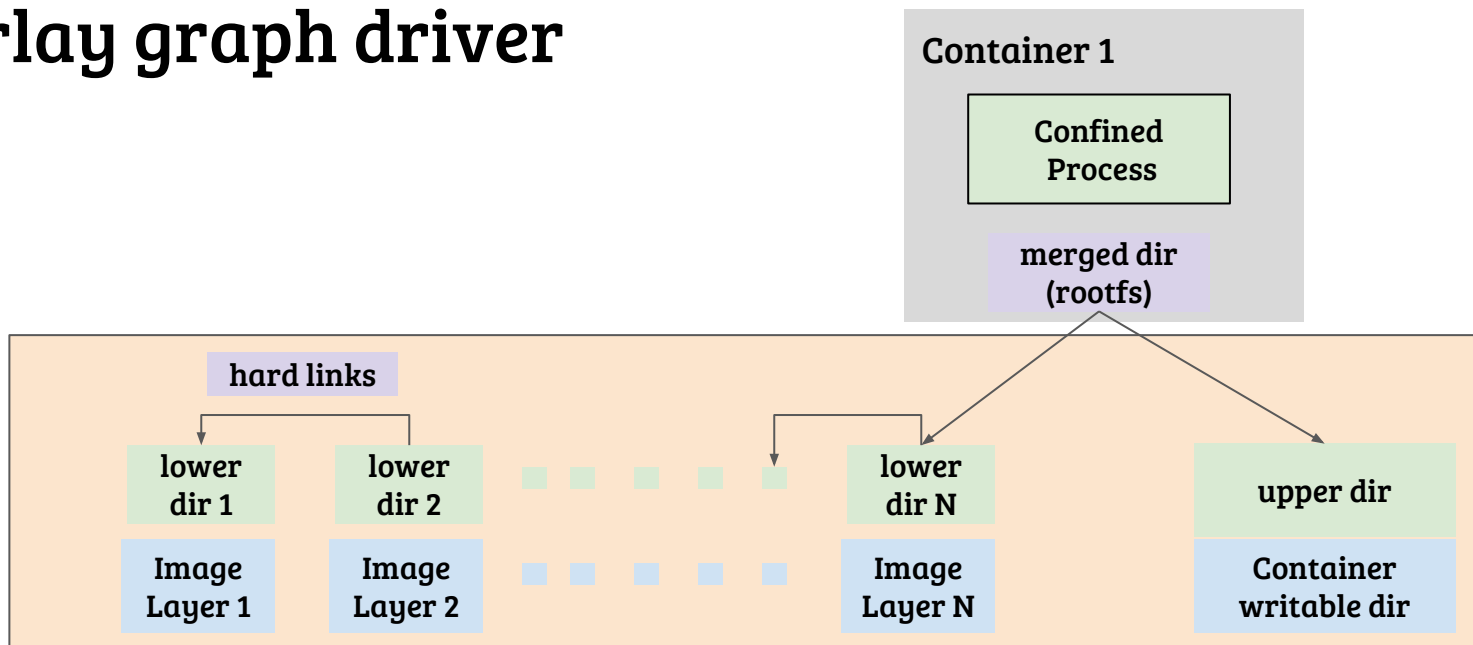
- Multiple lower layers
- Renaming directories
- SELinux
- POSIX ACL
- File locking

Features (work in progress)

- **RW-RO file consistency after copy-up**
 - Just need to fix this case up in VFS
- **Fix st_dev, constant st_ino/d_ino**
 - Store inode number for copied up files
 - Finding a common ino space for different underlying filesystems
- **Hard link copy up**
 - Should be very rare
 - Can use a global database for storing inode numbers of copied up hard links

overlays usage in docker

overlay graph driver



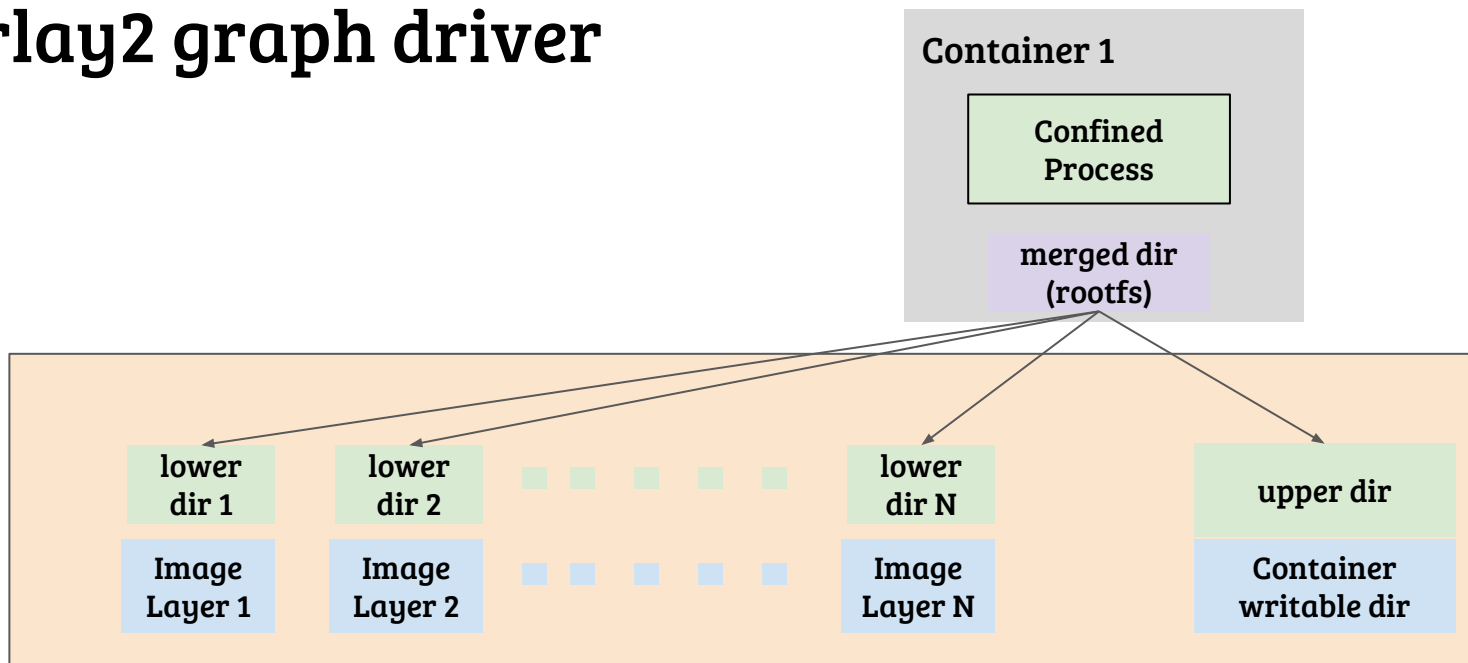
docker daemon option --storage-driver=overlay

Overlay supported single lower directory

Hard links created between image layers

Higher inode utilization

overlay2 graph driver



docker daemon option --storage-driver=overlay2

overlayfs should support multiple lower dirs

No hardlinks and dir creation in every layer

Better inode utilization

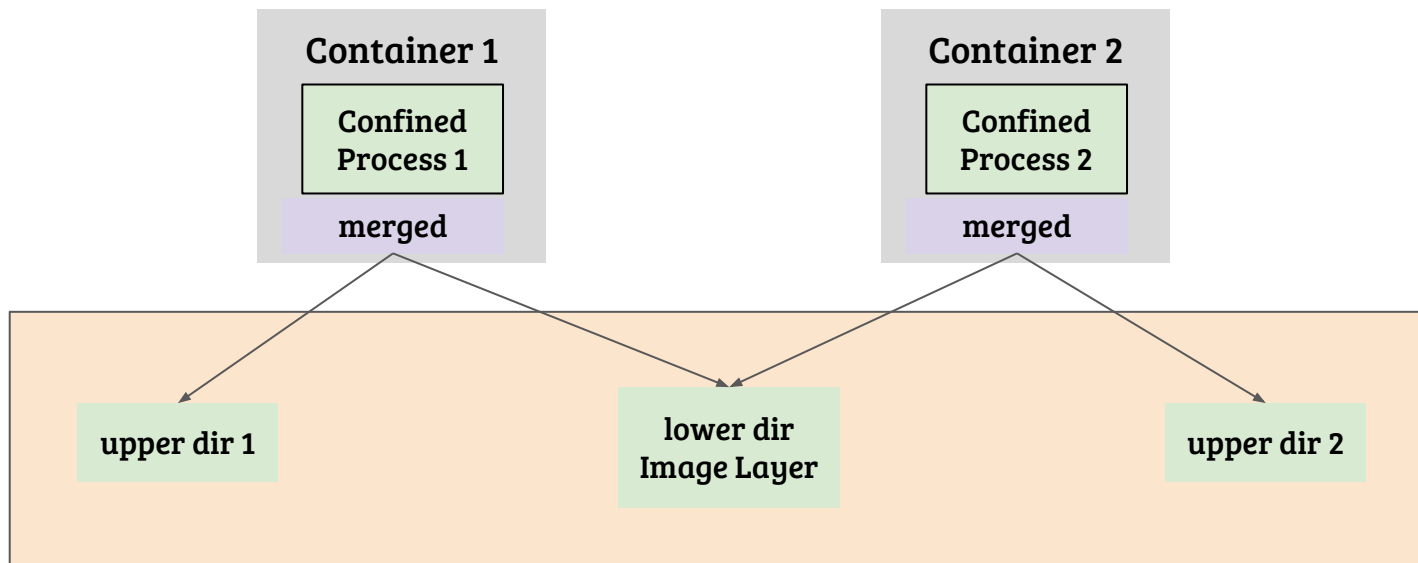
Container security and overlays

How do we handle access permissions?

DAC(Ownership/Permissions)

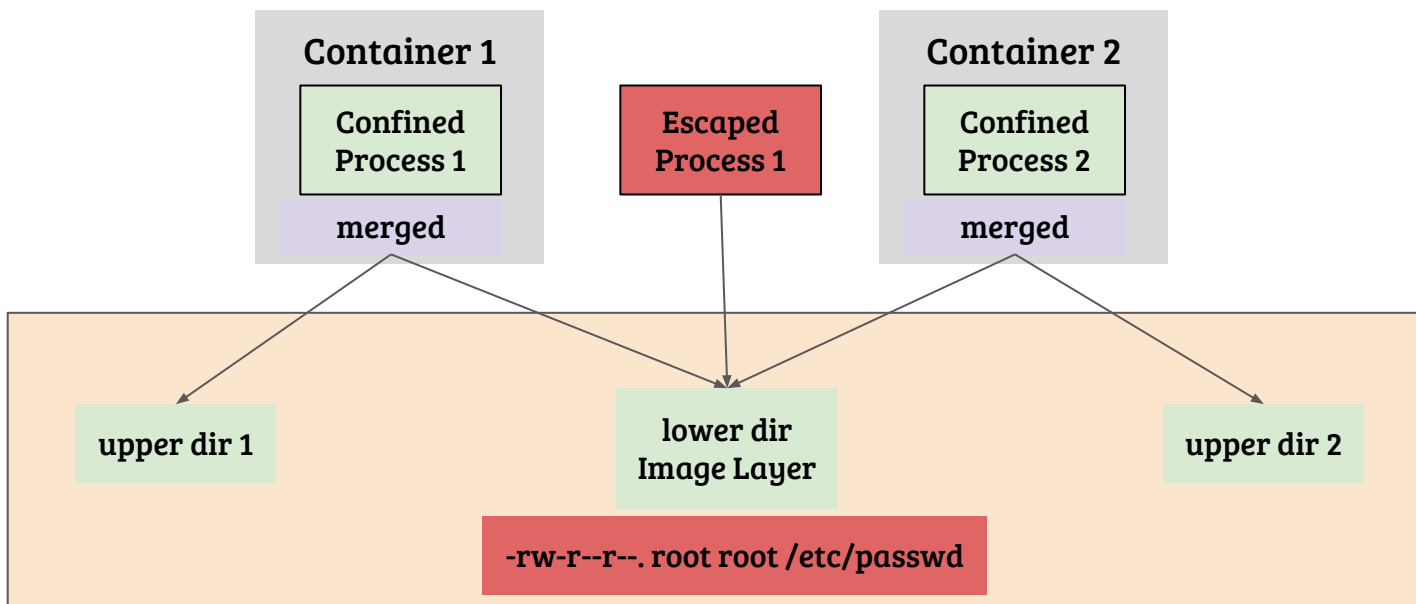
MAC (SELinux)

An example setup



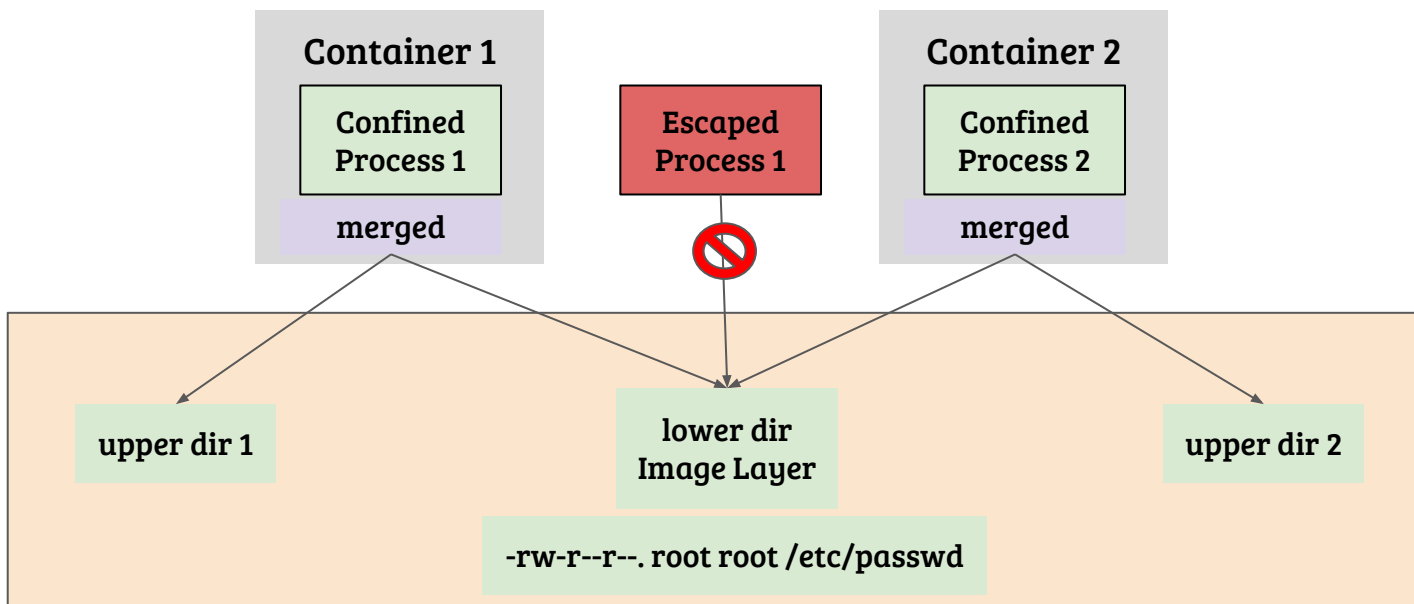
Two containers sharing lower dir with separate upper dir

Escaped container process writes to image dir/files



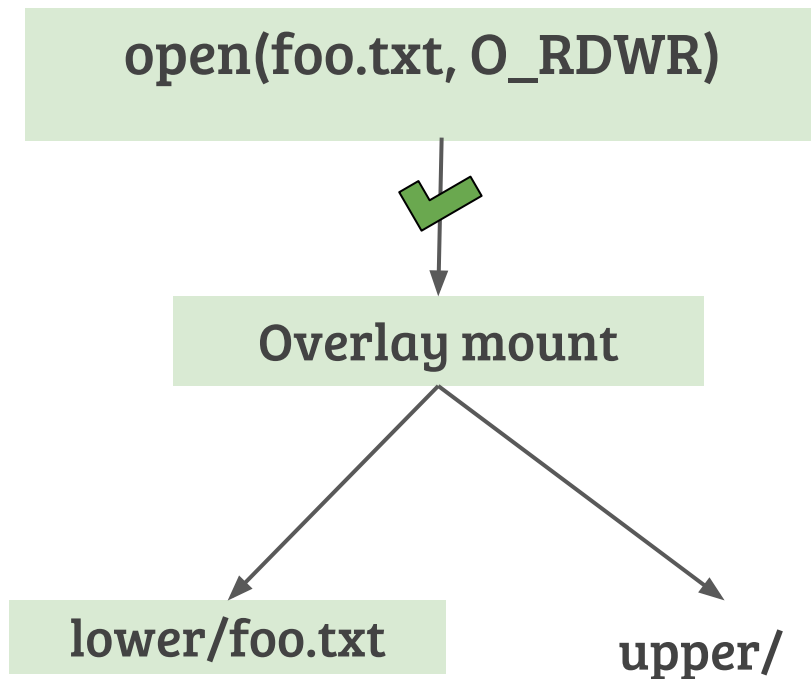
DAC allows writing to /etc/passwd

Security goal 1

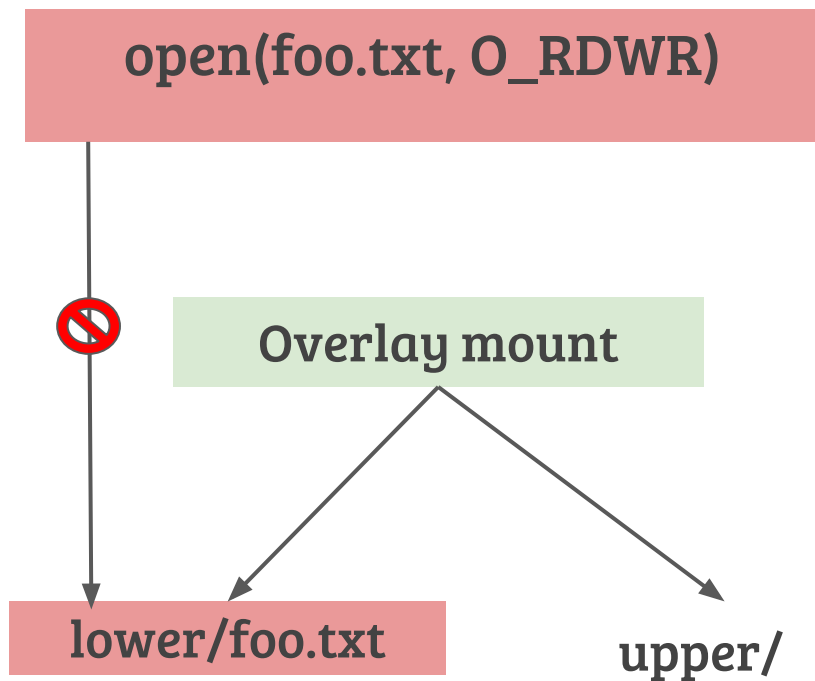


Do not allow writing to image dir/files

Allow access through overlay mount point

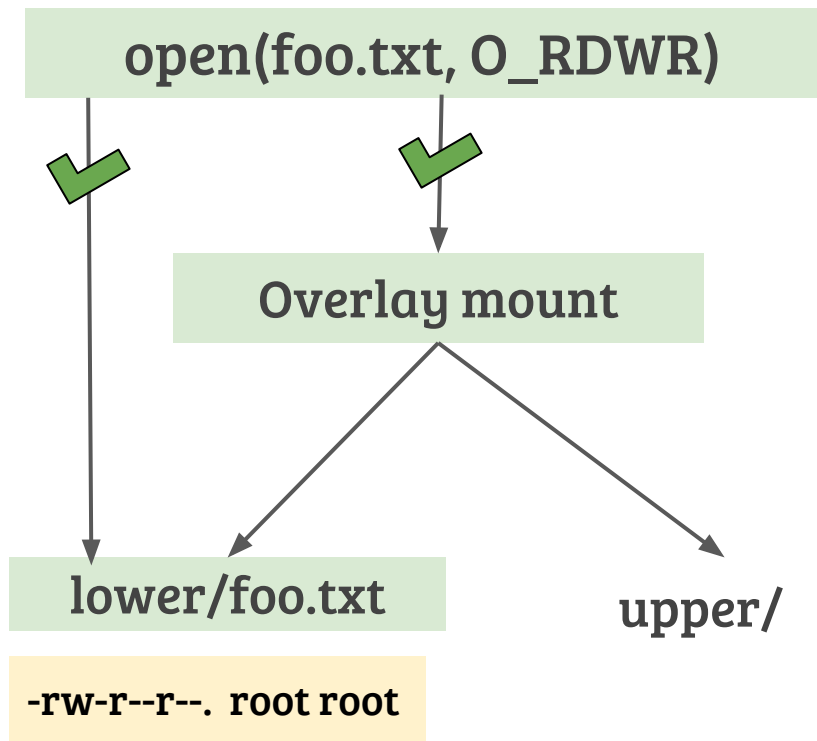


Deny write access on underlying file

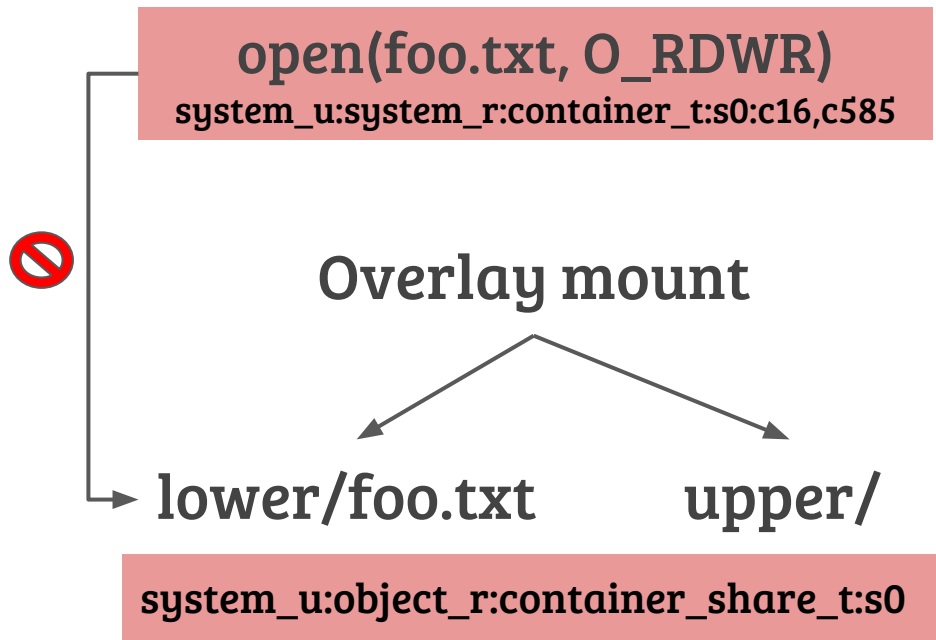


DAC allows access through both paths

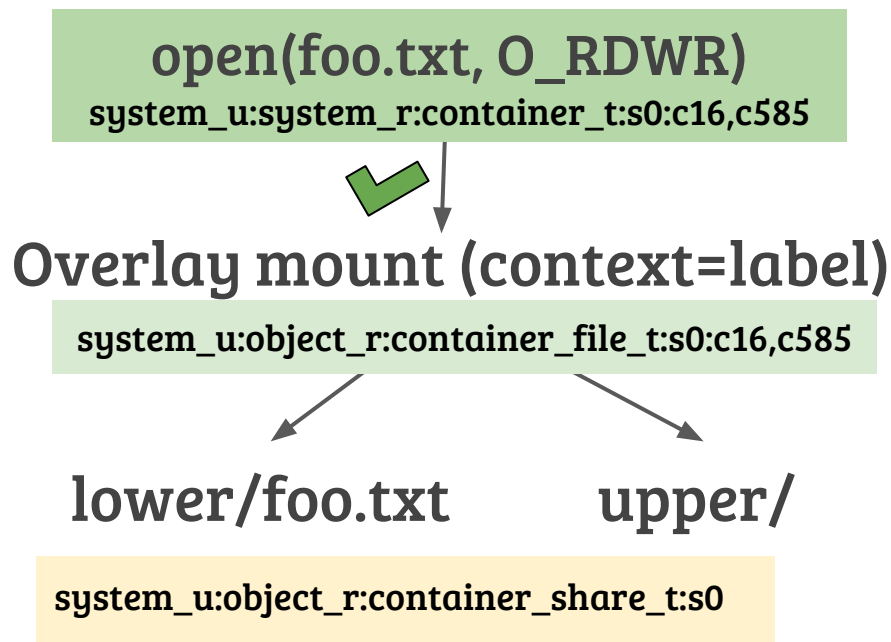
(When root inside container is root outside)



Read only label on lower files



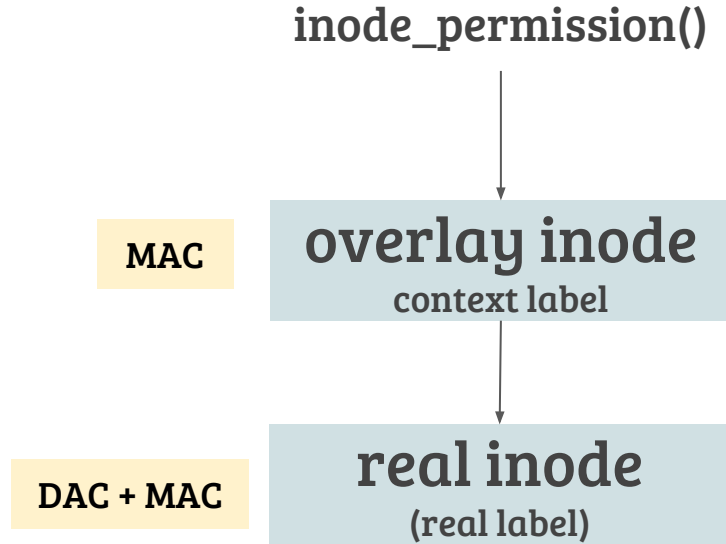
Use context mount option for overlay



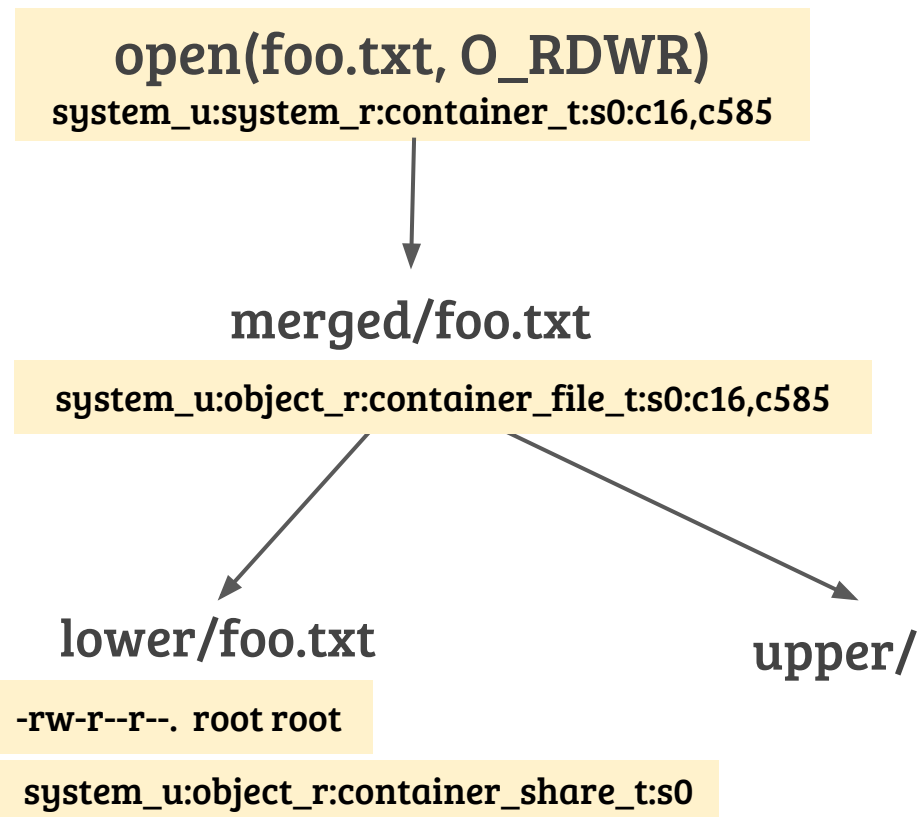
`mount -t overlay -o context="system_u:object_r:container_file_t:s0:c16,c585"... merged/`

That did not work

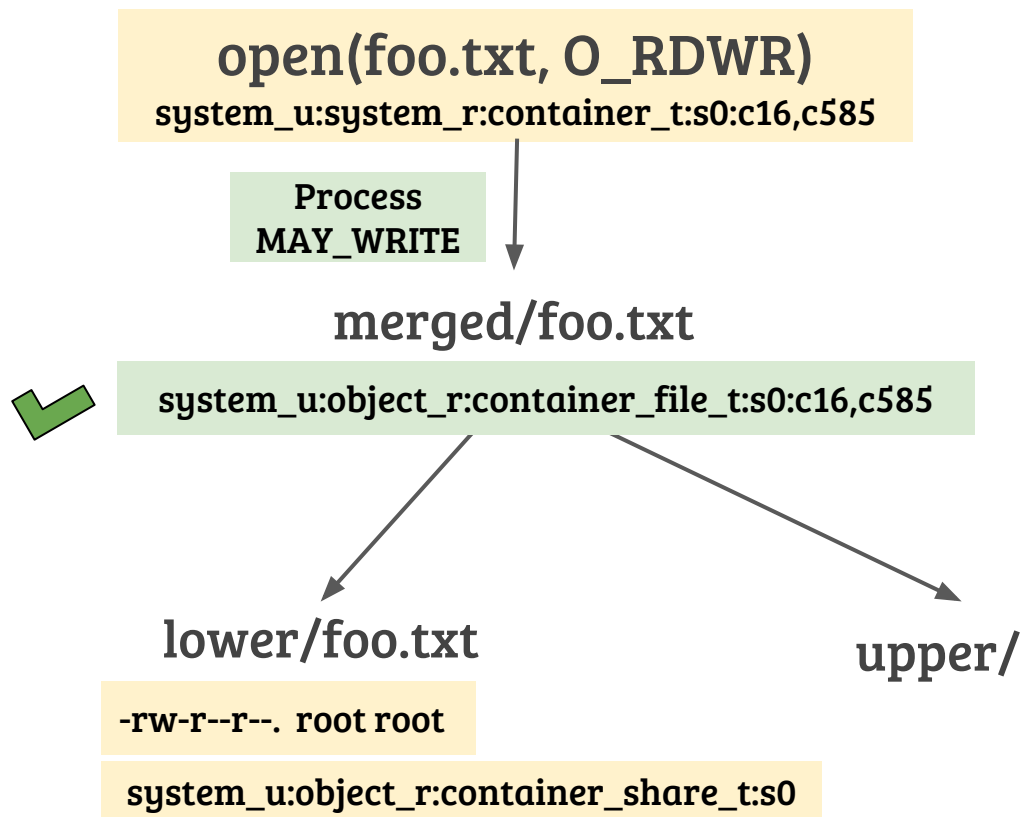
Access permission checks in overlay



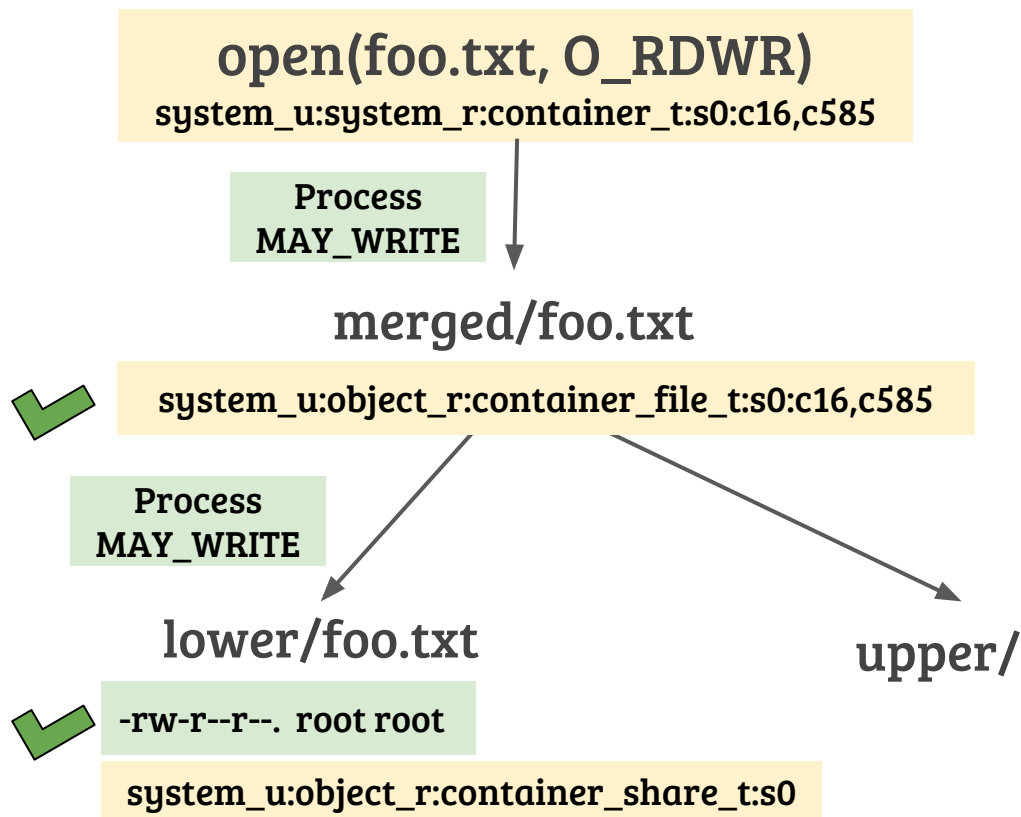
Read only label on lower file



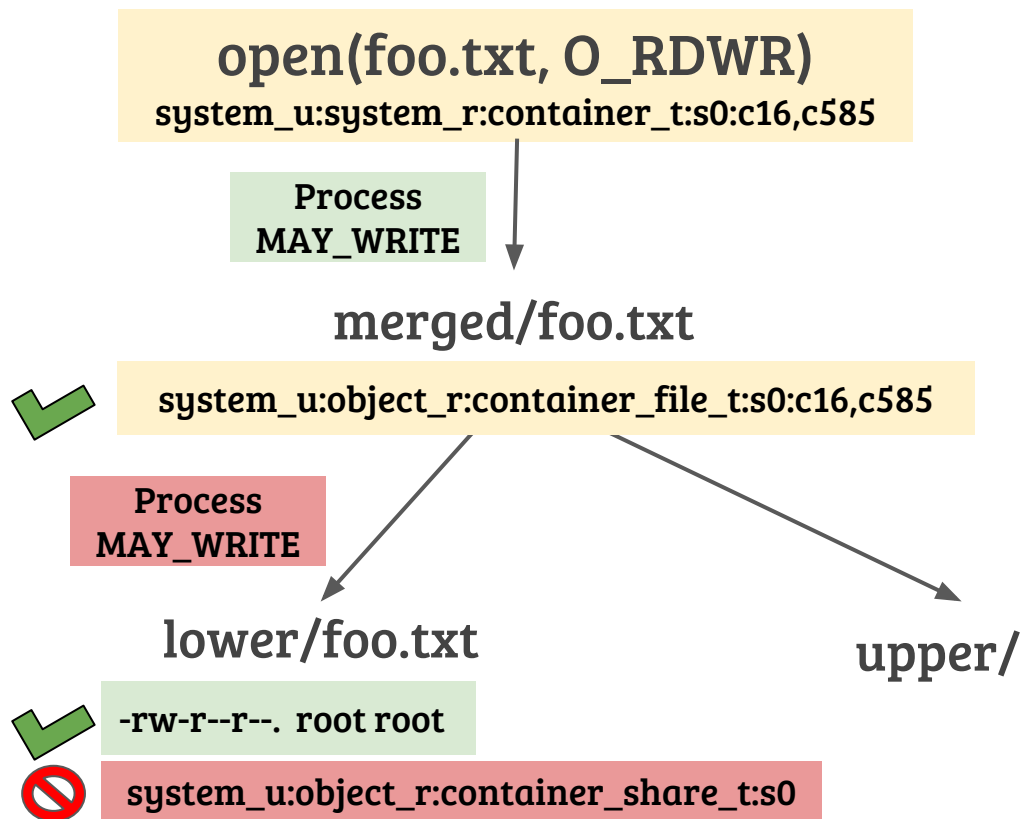
Process Overlay inode check



Process lower inode DAC check



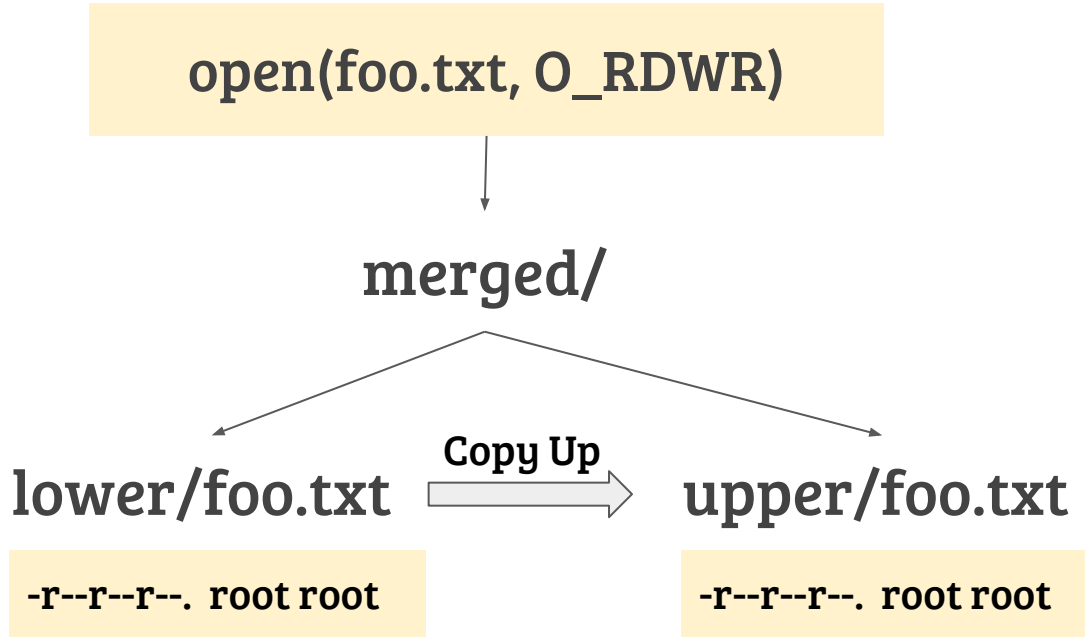
Process lower inode MAC check



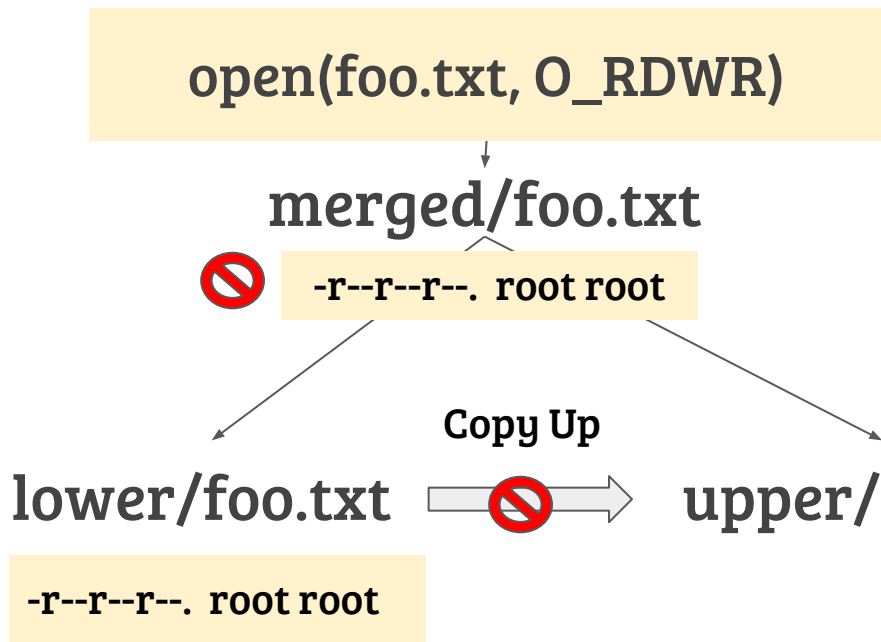
**What if we don't do WRITE checks on
lower inode**

But that will break DAC

DAC checks happen only at real inode



Why not do DAC checks on both inodes



That kind of worked but...

Certain overlays operations failed MAC checks

Certain overlays operations fail MAC checks

File creation over whiteout

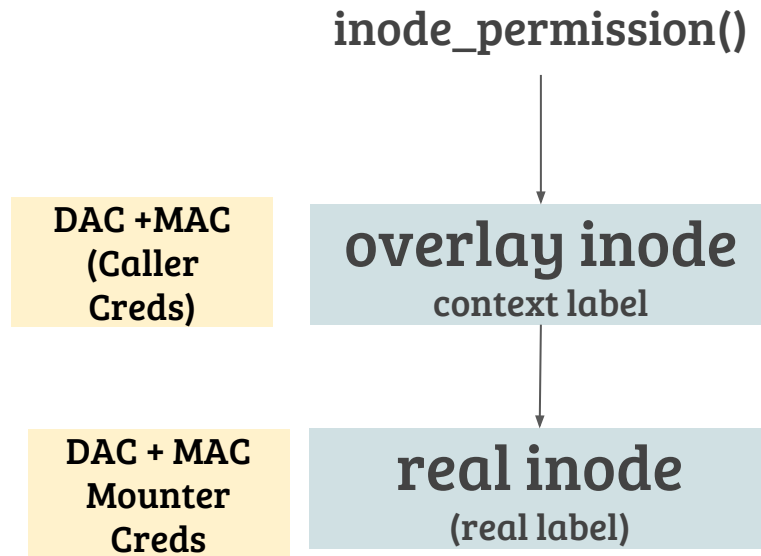
Certain overlays operations fail MAC checks

File creation over whiteout

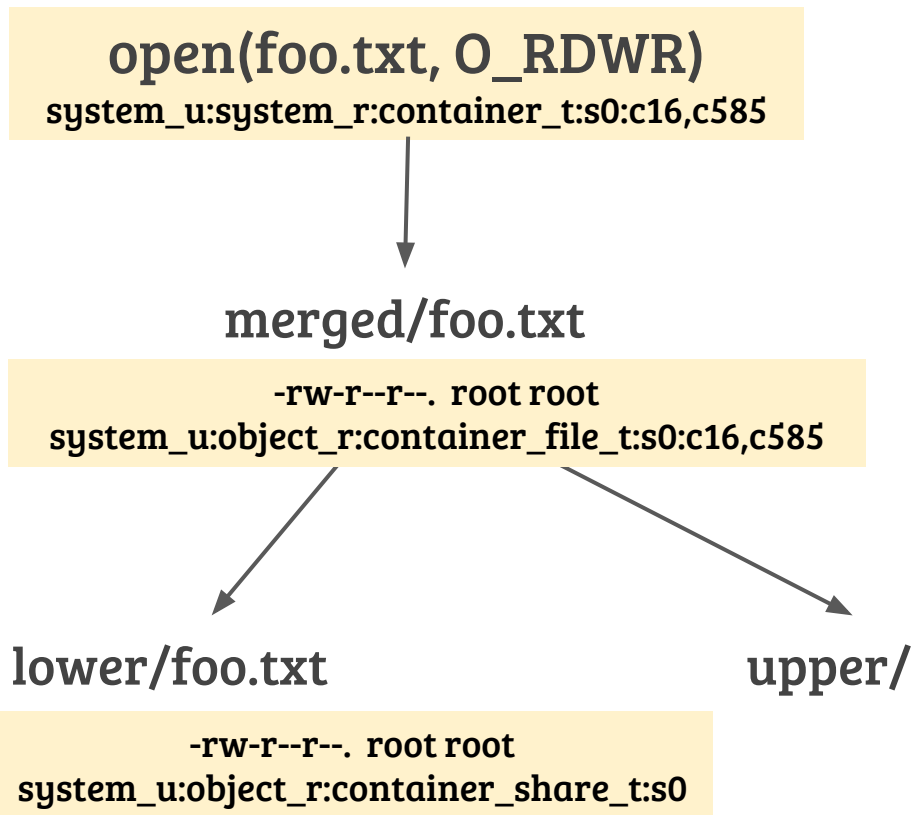
Use mounter's creds for privileged operations

Two Levels of Permission Checks

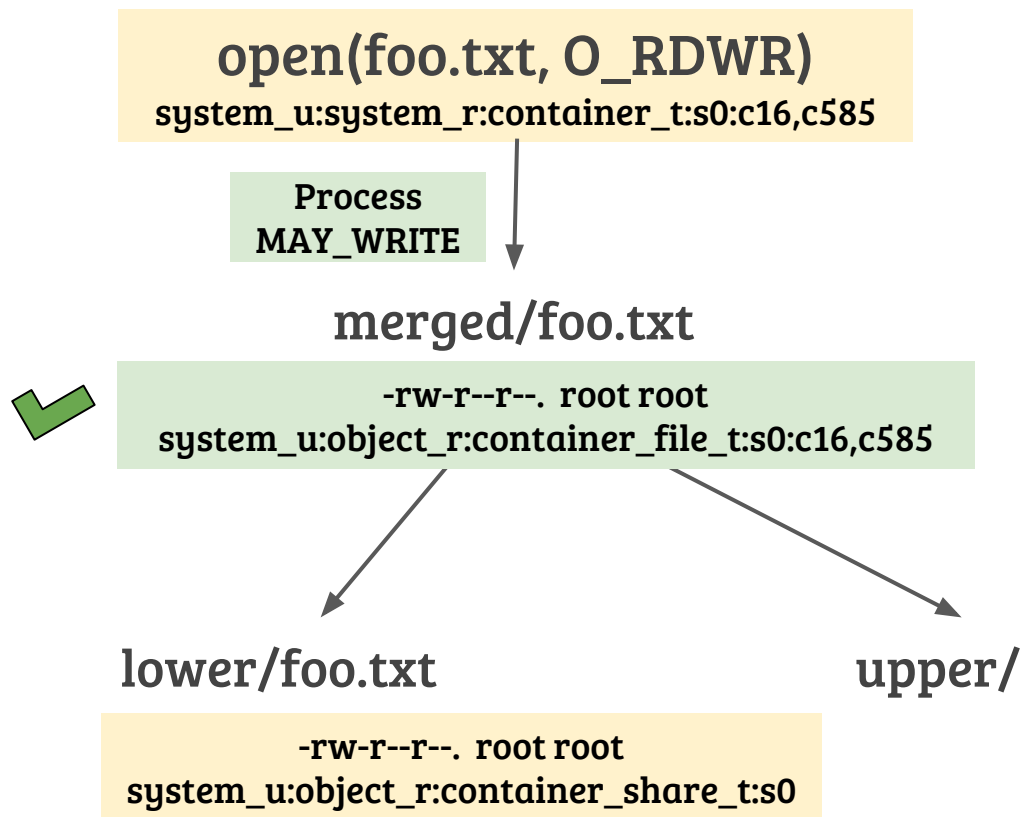
- Overlay inode is checked with creds of task
- Underlying inode is checked with creds of mounter
- Certain privileged operations are done with the creds of mounter



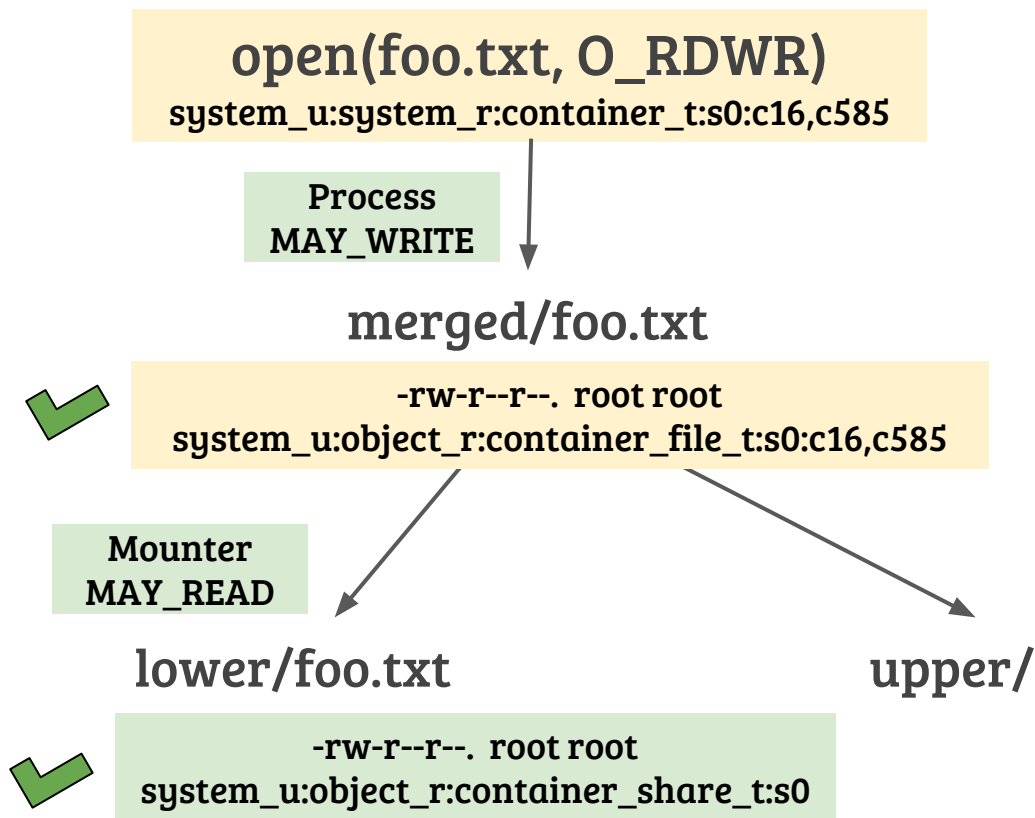
Two levels of checks



Process Overlay inode check

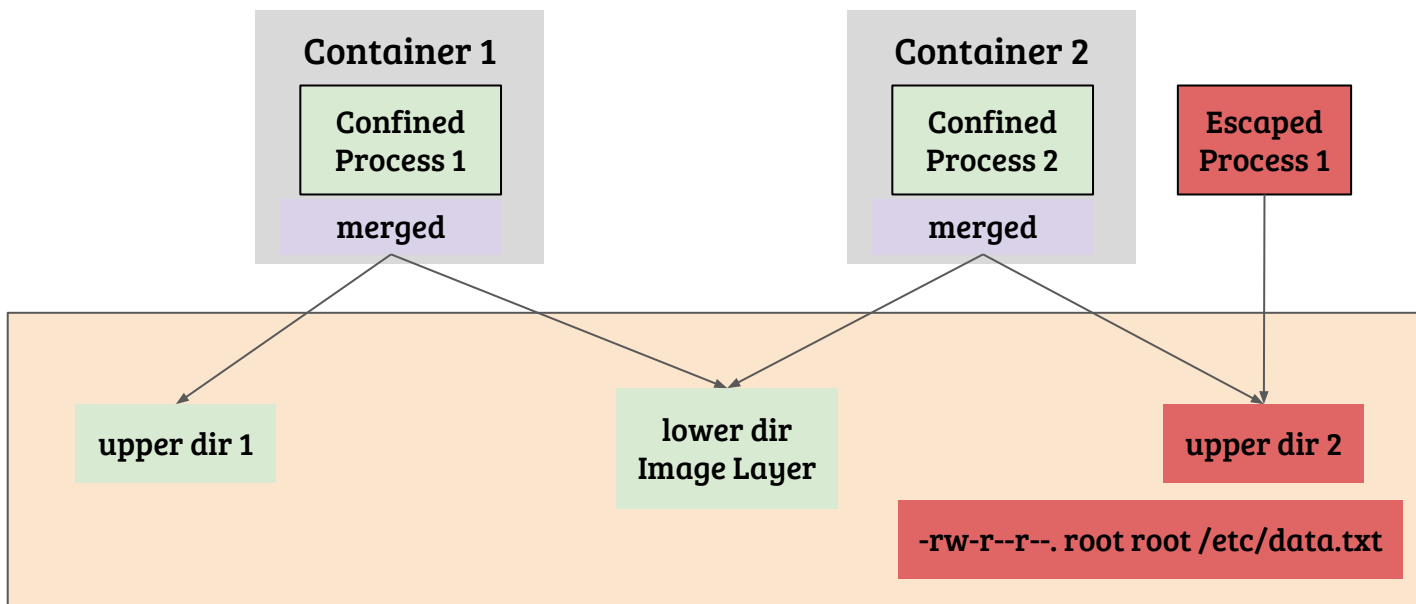


Mounter real lower inode check



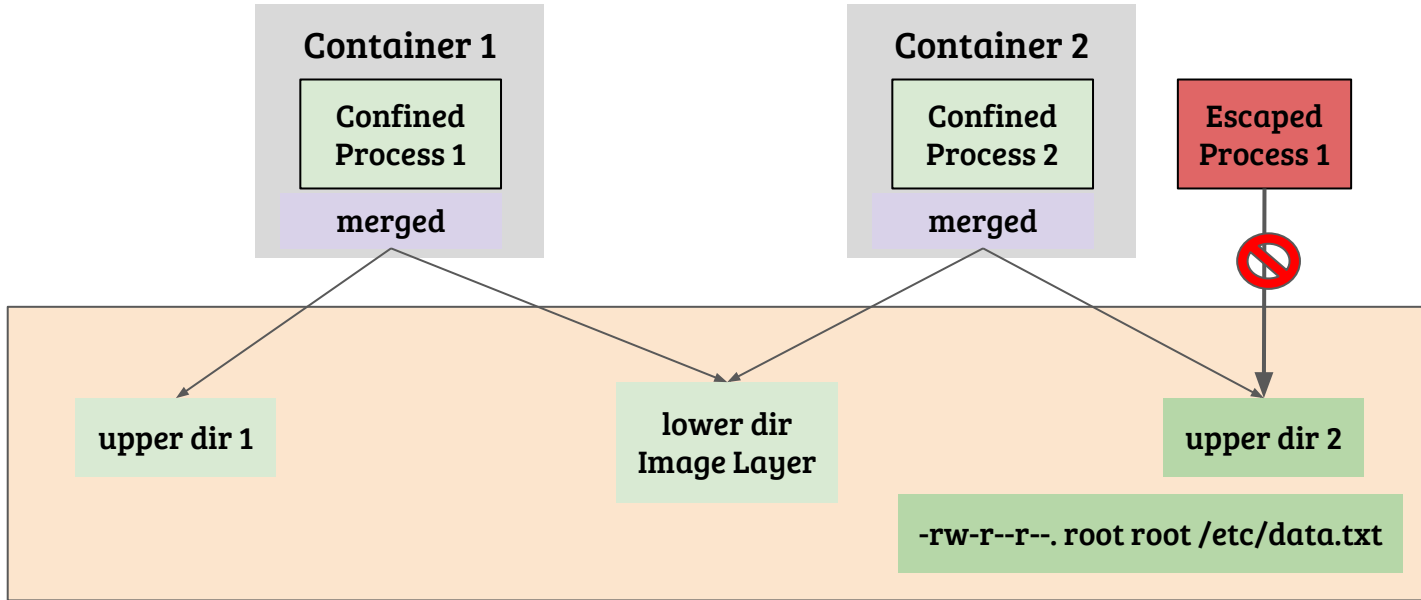
First requirement met

Escaped process accesses other container's data



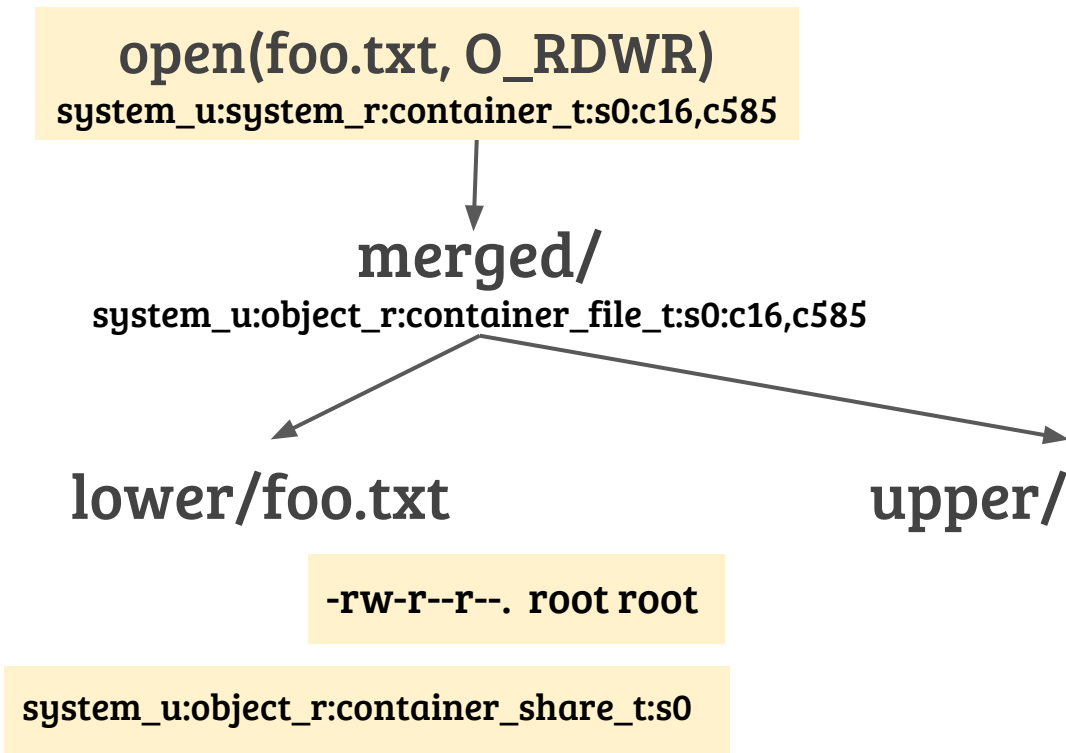
Container1 accesses container2's data

Security goal 2

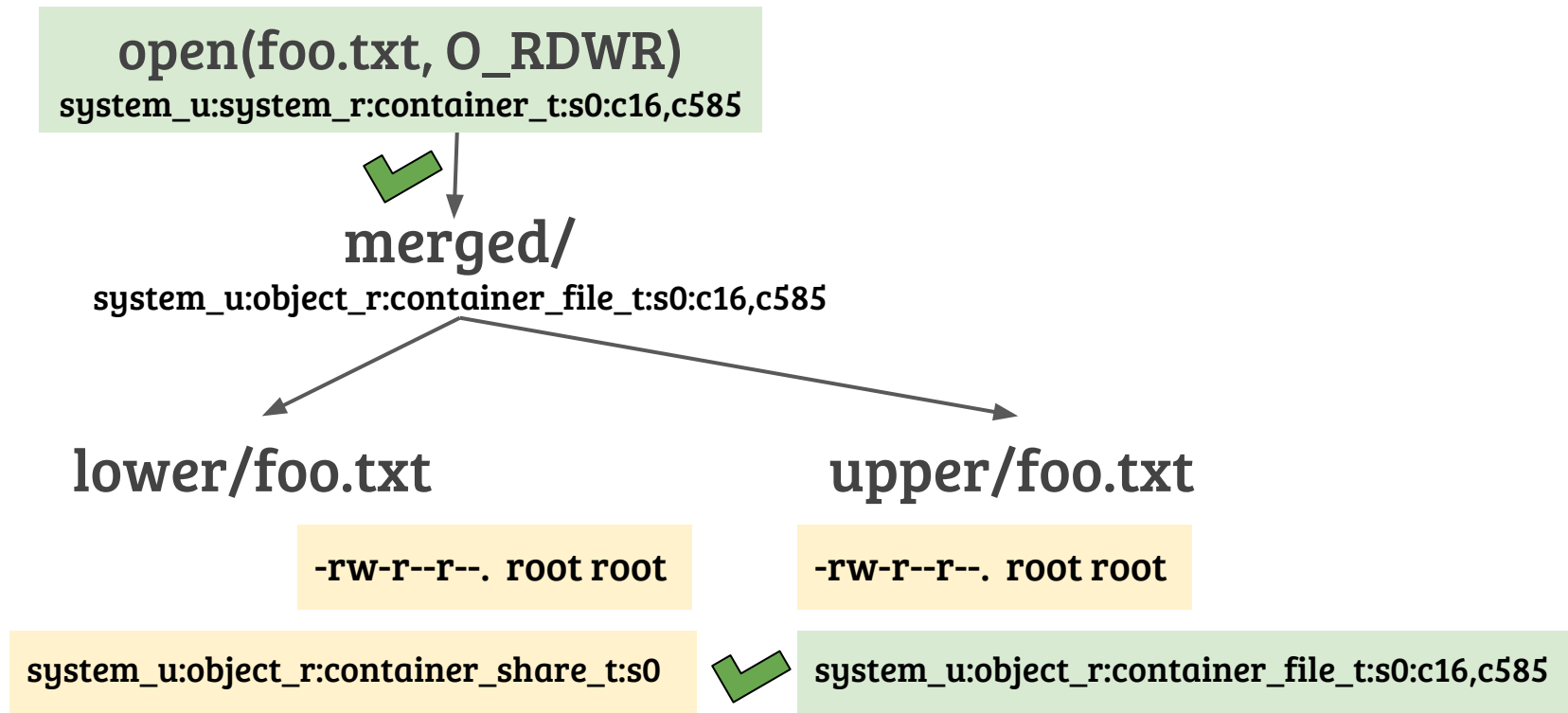


One container should not be able to access other container's data

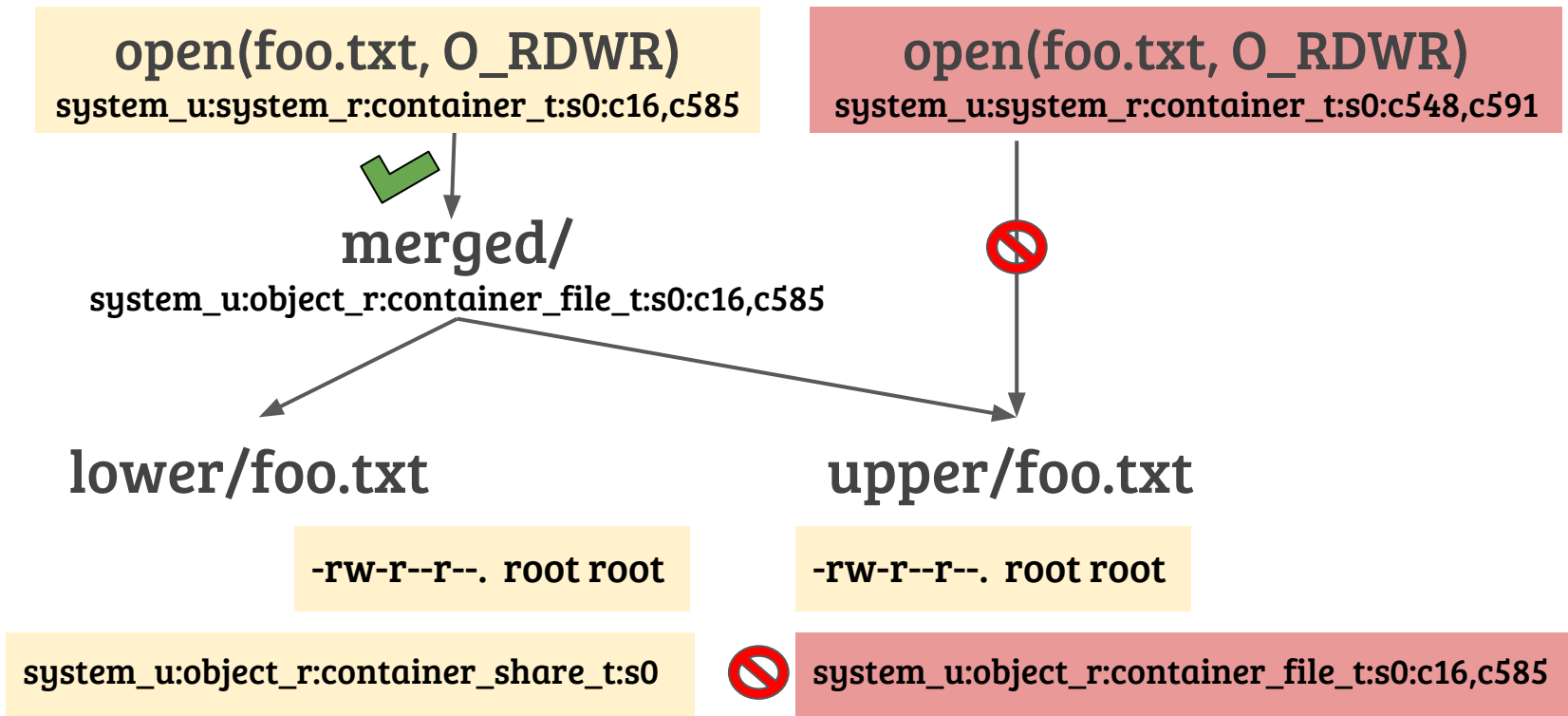
Label upper files for container access only



Label upper files for container access only



One container can't access data of another container



New LSM Hooks

- **inode_copy_up()**
 - Called during copy up. Returns new set of creds for file creation.
 - For context mounts, file is created with label specified in context= option.
- **inode_copy_up_xattr()**
 - Called during copy up of xattrs. SELinux blocks copying up of SELinux xattr.
- **dentry_create_files_as()**
 - Called during creation of new file. Returns new set of creds for file creation.
 - For context mounts, file is created with label specified in context= option.

Overlayfs vs. devicemapper

- **In general, faster than devicemapper**
 - Page cache sharing
- **Not fully POSIX compliant, yet**
 - So some workloads might experience issues
- **Fedora 26 will have overlay2 as default graph driver**
 - Switch back to devicemapper if you face issues

DAC and container security

- DAC will solve these issues if containers run in user namespaces with different mappings
- Needing to do a chown on image continues to be a issue
- shiftfs or something else?

Thank You