

MAC and DAC: Who Actually Decides Who Gets In

Sun, Aug 9, 2:00 PM EDT · <https://scottslab.io/posts/mac-and-dac-access-control-models>



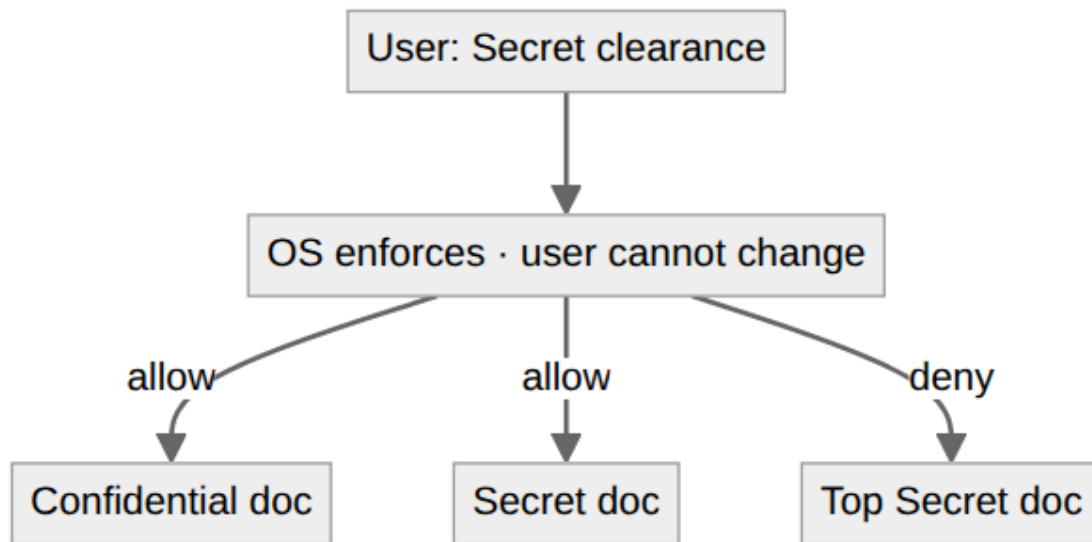
TL;DR

Access control models come down to who holds the pen. Under Mandatory Access Control the operating system enforces permissions from labels and users can't change them: think clearances and SELinux. Under Discretionary Access Control the resource owner grants and delegates access themselves. It's the everyday model, and exactly what NTFS permissions and ACLs on Windows are. MAC trades flexibility for rigor; DAC trades rigor for flexibility.

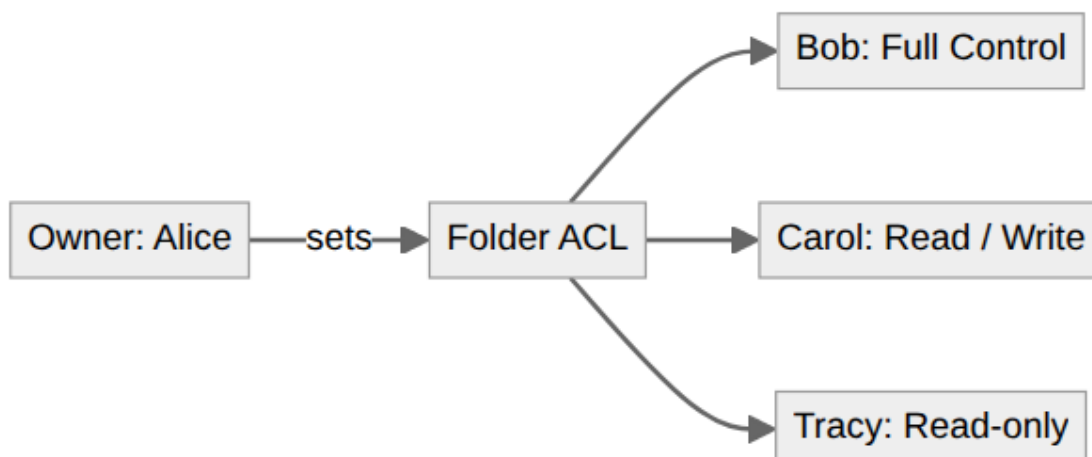
Every access control model is answering one question: who gets to decide what you can touch. The first split is between the *system* deciding and the *owner* deciding.

Mandatory Access Control puts the operating system in charge, and the user can't override it. It's rule-based in a precise sense: every user and every resource carries a label, and the OS compares them on each access. The canonical example is government clearances: a user cleared to Secret can open Secret and Confidential documents but not Top Secret, and nobody below can grant them that. The system enforces the lattice, full stop. On Linux the primary implementation is SELinux, originally built by the NSA and shipped in RHEL, Fedora, and the RHEL rebuilds like Rocky and AlmaLinux (CentOS lives on as CentOS Stream now that CentOS Linux is end-of-life). MAC is

rigid on purpose: it's what you want when a mistake, or a compromised account, must not be able to widen its own access.



Discretionary Access Control flips it: the resource owner sets the permissions and can delegate them. It's the most common model because it's flexible: the person who made the file decides who else gets in, which is how most organizations actually operate. On Windows this is NTFS, and its permission set is worth knowing by name: Full Control, Modify, Read & Execute, Read, and Write, each layering on more capability. In practice it's an ACL hanging off the object: Alice owns a folder and grants Bob Full Control, Carol read/write, and Tracy read-only, with no administrator in the loop.



The trade-off is the whole story. DAC's flexibility is also its weakness: owners over-share, misjudge, and forget to revoke, and there's no system-level backstop when they do. MAC's rigor is also its cost: it's heavier to administer and unforgiving, which is why you see it in high-assurance environments and in targeted enforcement (SELinux confining a specific service) rather than as the everyday default. Most shops run DAC and reach for MAC where the stakes justify taking the pen out of users' hands.

Written by Scott S. — Contact: scott@scottschlangen.com — scottslab.io — <https://scottslab.io/posts/mac-and-dac-access-control-models>