**Access Control in MAC-OS** Omkar Gavhane(2111MC08)

* MAC-OS supports security extensions based on the POSIX
* These security mechanisms include file system **Access Control Lists** and **Mandatory Access Control (MAC)**

**Access Control List(ACL)**

Access Control Lists (ACLs) extend the standard UNIX permission model in a POSIX compatible way.

This permits an administrator to take advantage of a more fine-grained permissions model.

The MAC-OS GENERIC kernel provides ACL support for UFS file systems. Users who prefer to compile a custom kernel must include the following option in their custom kernel configuration file:

options UFS\_ACL

If this option is not compiled in, a warning message will be displayed when attempting to mount a file system with ACL support. ACLs rely on extended attributes which are natively supported in UFS2.

### **Enabling ACL Support**

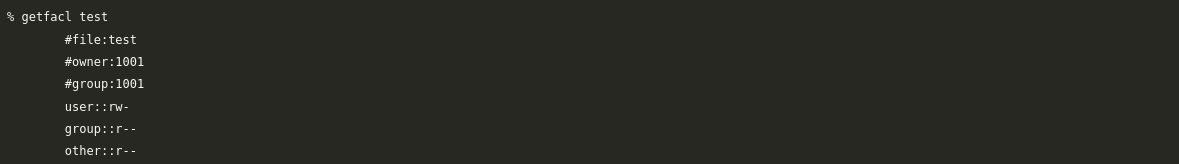
ACLs are enabled by the mount-time administrative flag,acls, which may be added to /etc/fstab

File systems with ACLs enabled will show a plus (+) sign in their permission settings:

In this example,directory1,directory2,and directory3 are all taking advantage of ACLs, whereas privateand public\_html are not.

### **Using ACLs**

File system ACLs can be viewed using getfacl. For instance, to view the ACL settings on **test**:



To change the ACL settings on this file, use setfacl. To remove all of the currently defined ACLs from a file or file system, include -k



To modify the default ACL entries, use -m:



**Mandatory Access Control(MAC)**

Mandatory Access Control (MAC) allows access control modules to be loaded in order to implement security policies

**Terminologies associated**

*level*: The increased or decreased setting of a security attribute. As the level increases, its security is considered to elevate as well.

*label*: A security attribute which can be applied to files, directories, or other items in the system. It could be considered a confidentiality stamp. When a label is placed on a file, it describes the security properties of that file and will only permit access by files, users, and resources with a similar security setting. The meaning and interpretation of label values depends on the policy configuration. Some policies treat a label as representing the integrity or secrecy of an object while other policies might use labels to hold rules for access.

*multilabel*: this property is a file system option which can be set in single-user mode using [tunefs](https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html), during boot using [fstab](https://www.freebsd.org/cgi/man.cgi?query=fstab&sektion=5&format=html), or during the creation of a new file system. This option permits an administrator to apply different MAC labels on different objects. This option only applies to security policy modules which support labeling.

*single label*: a policy where the entire file system uses one label to enforce access control over the flow of data. Whenever multilabel is not set, all files will conform to the same label setting.

*object*: an entity through which information flows under the direction of a subject. This includes directories, files, fields, screens, keyboards, memory, magnetic storage, printers or any other data storage or moving device. An object is a data container or a system resource. Access to an object effectively means access to its data

*subject*: any active entity that causes information to flow between objects such as a user, user process, or system process. On FreeBSD, this is almost always a thread acting in a process on behalf of a user.

**More about labels in Mandatory Access Control (MAC)**

A MAC label is a security attribute which may be applied to subjects and objects throughout the system

There are two types of label policies: *Single label and Multi label*.

By default, the system will use single label.

A single label security policy only permits one label to be used for every subject or object. Since a single label policy enforces one set of access permissions across the entire system, it provides lower administration overhead, but decreases the flexibility of policies which support labeling. However, in many environments, a single label policy may be all that is required.

A **single label policy** is somewhat similar to **Discretionary Access Control(DAC)** as root configures the policies so that users are placed in the appropriate categories and access levels. A notable difference is that many policy modules can also restrict root. Basic control over objects will then be released to the group, but root may revoke or modify the settings at any time.

When appropriate, a multi label policy can be set on a UFS file system by passing multilabel to [tunefs](https://www.freebsd.org/cgi/man.cgi?query=tunefs&sektion=8&format=html). A multi label policy permits each subject or object to have its own independent MAC label.

The decision to use a **multi label** or single label policy is only required for policies which implement the labeling feature, such as **biba,lomac, andmls**.

Some policies, such as **seeotheruids,portacl and partition, do not use labels** at all.

Using a multi label policy on a partition and establishing a multi label security model can increase administrative overhead as everything in that file system has a label. This includes directories, files, and even device nodes

command will set multilabel on the specified UFS file system is

tunefs -l enable

**Label configuration**

All configuration may be done using **setfmac**, which is used to **set MAC labels on system objects**, and **setpmac**, which is used to **set the labels on system subjects**.

For example, to set the biba MAC label to high on test command is

setfmac biba/high test

A few MAC-OS policy modules which support the labeling feature offer three predefined labels:**low**,**equal**, and **high**, where:

**low** is considered the lowest label setting an object or subject may have. Setting this on objects or subjects blocks their access to objects or subjects marked high.

**Equal** sets the subject or object to be disabled or unaffected and should only be placed on objects considered to be exempt from the policy.

**High** grants an object or subject the highest setting available in the Biba and MLS policy modules.

Such policy modules include [mac\_biba](https://www.freebsd.org/cgi/man.cgi?query=mac_biba&sektion=4&format=html),[mac\_mls](https://www.freebsd.org/cgi/man.cgi?query=mac_mls&sektion=4&format=html) and [mac\_lomac](https://www.freebsd.org/cgi/man.cgi?query=mac_lomac&sektion=4&format=html).

### **The MAC Multi-Level Security Module(MLS)**

Module name:**mac\_mls.ko**

Kernel configuration line:options MAC\_MLS

Boot option:mac\_mls\_load="YES"

The [mac\_mls](https://www.freebsd.org/cgi/man.cgi?query=mac_mls&sektion=4&format=html) policy controls access between subjects and objects in the system by enforcing a strict information flow policy

Three labels are included in this policy:mls/low,mls/equal, and mls/high, where:

* Anything labeled with mls/low will have a low clearance level and not be permitted to access information of a higher level. This label also prevents objects of a higher clearance level from writing or passing information to a lower level.
* mls/equal should be placed on objects which should be exempt from the policy.
* mls/high is the highest level of clearance possible. Objects assigned this label will hold dominance over all other objects in the system; however, they will not permit the leaking of information to objects of a lower class.

MLS provides:

* A hierarchical security level with a set of non-hierarchical categories.
* Fixed rules of no read up, no write down. This means that a subject can have read access to objects on its own level or below, but not above. Similarly, a subject can have write access to objects on its own level or above, but not beneath.
* Secrecy, or the prevention of inappropriate disclosure of data.
* A basis for the design of systems that concurrently handle data at multiple sensitivity levels without leaking information between secret and confidential.

To manipulate MLS labels, use [setfmac](https://www.freebsd.org/cgi/man.cgi?query=setfmac&sektion=8&format=html). To assign a label to an object:

setfmac mls/5 test

To get the MLS label for the file **test**

getfmac test

### **The MAC Biba Module**

Module name:**mac\_biba.ko**

Kernel configuration line:options MAC\_BIBA

Boot option:mac\_biba\_load="YES"

In Biba environments, an "integrity" label is set on each subject or object

Supported labels are biba/low,biba/equal, and biba/high, where:

* biba/low is considered the lowest integrity an object or subject may have. Setting this on objects or subjects blocks their write access to objects or subjects marked as biba/high, but will not prevent read access.
* biba/equal should only be placed on objects considered to be exempt from the policy.
* biba/high permits writing to objects set at a lower label, but does not permit reading that object. It is recommended that this label be placed on objects that affect the integrity of the entire system.

Biba provides:

* Hierarchical integrity levels with a set of non-hierarchical integrity categories.
* Fixed rules are no write up, no read down, the opposite of MLS. A subject can have write access to objects on its own level or below, but not above. Similarly, a subject can have read access to objects on its own level or above, but not below.
* Integrity by preventing inappropriate modification of data.
* Integrity levels instead of MLS sensitivity levels.

To access the Biba policy setting on system objects, use setfmac and getfmac:

setfmac biba/low test

getfmac test

test:biba/low