# Security Model

DAOS uses a flexible security model that separates authentication from authorization. It is designed to have a minimal impact on the I/O path.

DAOS does not provide any transport security for the fabric network used for I/O transfers. When deploying DAOS, the administrator is responsible for secure configuration of their specific fabric network. For RDMA over Ethernet, enabling IPsec is recommended. See the [RDMA protocol spec (RFC 5040)](https://tools.ietf.org/html/rfc5040#section-8.2) for more information.

There are two areas where DAOS implements its own layer of security. At the user level, clients must be able to read and modify only *pools* and *containers* to which they have been granted access. At the system and administrative levels, only authorized components must be able to access the DAOS management network.

## Authentication

There are different means of authentication, depending on whether the caller is accessing client resources or the DAOS management network.

### Client Library

The client library libdaos is an untrusted component. The daos user-level command that uses the client library is also an untrusted component. A trusted process, the DAOS agent (daos\_agent), runs on each client node and authenticates the user processes.

The DAOS security model is designed to support different authentication methods for client processes. Currently, we support AUTH\_SYS authentication only.

### DAOS Management Network

Each trusted DAOS component (daos\_server, daos\_agent, and the dmg administrative tool) is authenticated by means of a certificate generated for that component. These components identify one another over the DAOS management network via mutually-authenticated TLS.

## Authorization

Client authorization for resources is controlled by the Access Control List (ACL) on the resource. Authorization on the management network is achieved by settings on the [certificates](https://daos-stack.github.io/admin/deployment/#certificate-configuration) that are generated while setting up the DAOS system.

### Component Certificates

Access to DAOS management RPCs is controlled via the CommonName (CN) set in each management component certificate. A given management RPC may only be invoked by a component which connects with the correct certificate.

### Access Control Lists

Client access to resources like *pools* and *containers* is controlled by DAOS Access Control Lists (ACLs). These ACLs are derived in part from NFSv4 ACLs, and adapted for the unique needs of a distributed system.

The client may request read-only or read-write access to the resource. If the resource ACL doesn’t grant them the requested access level, they won’t be able to connect. While connected, their handle to that resource grants permissions for specific actions.

The permissions of a handle last for the duration of its existence, similar to an open file descriptor in a POSIX system. A handle cannot currently be revoked.

#### Access Control Entries

In the input and output of DAOS tools, an Access Control Entry (ACE) is defined using a colon-separated string with the following format: TYPE:FLAGS:PRINCIPAL:PERMISSIONS

The contents of all the fields are case-sensitive.

##### Type

The type of ACE entry (mandatory). Only one type of ACE is supported at this time.

* A (Allow): Allow access to the specified principal for the given permissions.

##### Flags

The (optional) flags provide additional information about how the ACE should be interpreted.

* G (Group): The principal should be interpreted as a group.

##### Principal

The principal (also called the identity) is specified in the name@domain format. The domain should be left off if the name is a UNIX user/group on the local domain. Currently, this is the only case supported by DAOS.

There are three special principals, OWNER@, GROUP@, and EVERYONE@, which align with User, Group, and Other from traditional POSIX permission bits. When providing them in the ACE string format, they must be spelled exactly as written here, in uppercase with no domain appended. The GROUP@ entry must also have the G (group) flag.

##### Permissions

The permissions in a resource’s ACE permit a certain type of user access to the resource. The order of the permission “bits” (characters) within the PERMISSIONS field of the ACE is not significant.

|  |  |  |
| --- | --- | --- |
| Permission | Pool Meaning | Container Meaning |
| r (Read) | Alias for ‘t’ | Read container data and attributes |
| w (Write) | Alias for ‘c’ + ‘d’ | Write container data and attributes |
| c (Create) | Create containers | N/A |
| d (Delete) | Delete any container | Delete this container |
| t (Get-Prop) | Connect/query | Get container properties |
| T (Set-Prop) | N/A | Set/Change container properties |
| a (Get-ACL) | N/A | Get container ACL |
| A (Set-ACL) | N/A | Set/Change container ACL |
| o (Set-Owner) | N/A | Set/Change container’s owner user and group |

ACEs containing permissions not applicable to the given resource are considered invalid.

To allow a user/group to connect to a resource, that principal’s permissions must include at least some form of read access (for example, read or get-prop). A user with write-only permissions will be rejected when requesting RW access to a resource.

##### Denying Access

Currently, only “Allow” Access Control Entries are supported.

However, it is possible to deny access to a specific user by creating an Allow entry for them with no permissions. This is fundamentally different from removing a user’s ACE, which allows other ACEs in the ACL to determine their access.

It is *not* possible to deny access to a specific group in this way, due to [the way group permissions are enforced](#enforcement).

##### ACE Examples

* A::daos\_user@:rw
  + Allow the UNIX user named daos\_user to have read-write access.
* A:G:project\_users@:tc
  + Allow anyone in the UNIX group project\_users to access a pool’s contents and create containers.
* A::OWNER@:rwdtTaAo
  + Allow the UNIX user who owns the container to have full control.
* A:G:GROUP@:rwdtT
  + Allow the UNIX group that owns the container to read and write data, delete the container, and manipulate container properties.
* A::EVERYONE@:r
  + Allow any user not covered by other rules to have read-only access.
* A::daos\_user@:
  + Deny the UNIX user named daos\_user any access to the resource.

#### Enforcement

Access Control Entries (ACEs) will be enforced in the following order:

* Owner-User
* Named users
* Owner-Group and named groups
* Everyone

In general, enforcement will be based on the first match, ignoring lower-priority entries.

If the user is the owner of the resource and there is an OWNER@ entry, they will receive the owner permissions only. They will not receive any of the permissions in the named user/group entries, even if they would match those other entries.

If the user isn’t the owner, or there is no OWNER@ entry, but there is an ACE for their user identity, they will receive the permissions for their user identity only. They will not receive the permissions for any of their groups, even if those group entries have broader permissions than the user entry does. The user is expected to match at most one user entry.

If no matching user entry is found, but entries match one or more of the user’s groups, enforcement will be based on the union of the permissions of all matching groups, including the owner-group GROUP@.

If no matching groups are found, the EVERYONE@ entry’s permissions will be used, if it exists.

By default, if a user matches no ACEs in the ACL list, access will be denied.

#### ACL File

Tools that accept an ACL file expect it to be a simple text file with one ACE on each line. A line may be marked as a comment by using a # as the first non-whitespace character on the line.

For example:

# ACL for my container  
# Owner can't touch data - just do admin-type things  
A::OWNER@:dtTaAo  
# My project's users can generate and access data  
A:G:my\_great\_project@:rw  
# Bob can use the data to generate a report  
A::bob@:r

The permission bits and the ACEs themselves don’t need to be in any specific order. However the order may be different when the resulting ACL is parsed and displayed by DAOS.

#### Limitations

The maximum size of the ACE list in a DAOS ACL internal data structure is 64KiB.

To calculate the internal data size of an ACL, use the following formula for each ACE:

* The base size of an ACE is 256 Bytes.
* If the ACE principal is *not* one of the special principals:
  + Add the length of the principal string + 1.
  + If that value is not 64-Byte aligned, round up to the nearest 64-Byte boundary.