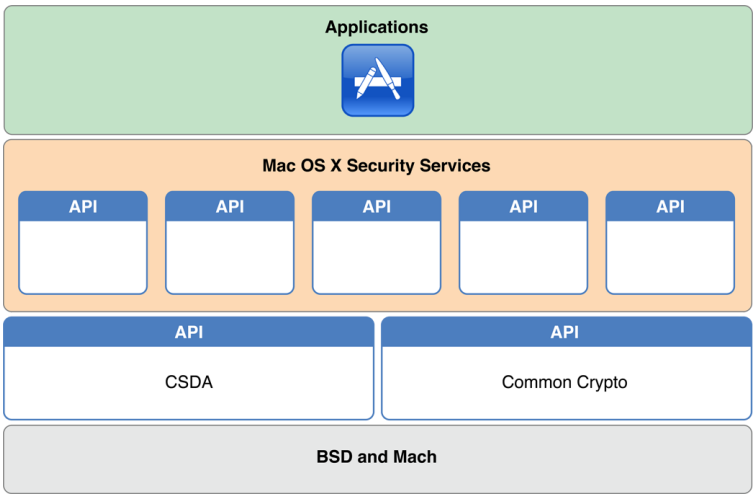
About Authentication, Authorization, and Permissions

Authentication and authorization are a key aspect of computer security. *Authentication* means determining the identity of a user, server, or client. *Authorization* means determining whether that user, server, or client as permission to do something. *Permissions* are settings on a file or other object that define who or what is allowed to use it and what they are allowed to do with it. This document describes the way macOS approaches authentication, authorization, and permissions.

Note: Authentication, authorization, and permissions are mostly relevant in the context of multiuser systems. As a result, this document only discusses macOS.



At a Glance

As you saw in *Security Overview*, macOS provides a wide range of security technologies that you can use when securing your application and its data. Authentication, authorization, and permissions in macOS are largely based on two open-source standards: Mach and BSD. These technologies sit at the lowest layer of macOS. The cryptography used for storing passwords and other secret information is provided by higher level technologies, such as Common Crypto. Common Crypto is an Apple open source technology that provides support for symmetric and asymmetric encryption, hashing, and other cryptography-related tasks.

Historical Note: Many macOS security services were originally built on top of CDSA, but that is now deprecated. See *Cryptographic Services Guide* for information about CDSA.

On top of these three technologies, macOS layers a number of security APIs (most of which are in the Core Services layer), including Security Transforms, the Security Interface framework, and Keychain Services. The Security Objective-C API is described in this document. The others are described in *Cryptographic Services Guide*.

macOS Provides Many Authentication and Identification Schemes and Technologies

There are many different ways to do authentication and identification, and macOS provides a number of authentication-related APIs and UI elements to help you. macOS also provides a number of authentication features to help you integrate macOS into enterprise environments.

Relevant Chapter: Authentication and Identification In Depth

Authorization Services Provides Centralized Management of Privileges

The Authorization Services API lets you run privileged helpers, and lets you use a single, central source of information for limiting what features of your app are usable by different accounts on the system. The Authorization Services API is not supported in sandboxed apps.

Relevant Chapter: Using Authorization

Note: Writing authorization plug-ins is beyond the scope of this document. To learn more about writing these plug-ins, read *Running At Login*.

Permissions and Access Control Limit Program Behavior

macOS supports permissions systems at various levels of the system, from Mach ports to file system permissions and mandatory access control.

The BSD portions of macOS provide fundamental services, including a user and group identification scheme, file system security policies based on users and groups, and network security policies.

The Mach portions of the macOS kernel provide fundamental services, including memory management, process management, thread management, hardware abstraction (with the help of the I/O Kit), and Mach port-based communication. Mach enforces access by controlling which tasks can send a message to a given Mach port, where a Mach port represents a task or some other Mach resource.

Relevant Chapter: Understanding Permissions

Prerequisites

Before reading this document, you should be familiar with the concepts in *Security Overview* and *Secure Coding Guide*.

See Also

For more information on related technologies, consider the following documents:

- *Security Overview*—Provides an overview of security concepts
- *Cryptographic Services Guide*—Describes the cryptographic features of macOS and iOS
- *Running At Login*—Describes how to write authorization services plug-ins for adding authentication schemes
- *Secure Coding Guide*—Explains how to write code that is robust against security holes