Security Server and Security Agent

Published: 2012-12-13 · Archived: 2026-04-05 18:58:07 UTC

The macOS and iOS security implementation includes a daemon called the *Security Server* that implements several security protocols, such as access to keychain items and root certificate trust management. macOS also includes a separate per-user agent, called the *Security Agent*, that is used by the Security Server to display a user interface.

This appendix briefly describes their roles.

Security Server

The *Security Server* (`securityd`) is a daemon running in macOS and iOS that implements several security protocols, such as encryption, decryption, and (in macOS) authorization computation.

In macOS and iOS, the Security Server listens for messages from various security APIs and performs cryptographic services on their behalf. Because developers generally use references to keys rather than using the keys themselves, the Security Server can keep those keys in a separate address space from the client process, thus reducing the risk of accidental disclosure.

As an added advantage, whenever Apple introduces new authentication or encryption technology, existing software that uses the macOS security APIs can transparently support it without code changes, provided that the software does not need to import or export keys directly.

The Security Server has no public API. Instead, your code calls APIs such as Keychain Services; Certificate, Key, and Trust Services; and Authorization Services (only on macOS), which in turn communicate with the Security Server.

Security Agent

The *Security Agent* is a separate process that provides the user interface for the Security Server in macOS (not iOS). Its primary purpose is to request authentication whenever an app requests additional privileges.

When the Security Server requires the user to authenticate, the Security Agent displays a dialog requesting a user name and password. The advantages of performing this action in a separate process are twofold. First, an application can obtain authorization without ever having access to the user’s credentials (username and password, for example). Second, it enables Apple to add new forms of authentication without requiring every application to understand them.

The Security Agent requires that the user be physically present in order to be authenticated. Because the graphical user interface elements can’t be used through a command-line interface such as the Terminal app or a secure shell (`ssh`) remote session, this restriction makes it much more difficult for a malicious user to breach an app’s security.
