Garantir

GaraSign macOS User Guide

Version 1.25.0

Table of Contents

Preface

Document Information

Title	GaraSign macOS User Guide
Product Version	1.25.0
Release Date	April 2024

Trademarks

Disclaimer

Garantir makes no representations or warranties with respect to the contents of this document and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Furthermore, Garantir reserves the right to revise this publication and to make changes from time to time in the content hereof without the obligation upon Garantir to notify any person or organization of any such revisions or changes.

We have attempted to make these documents complete, accurate, and useful, but we cannot guarantee them to be without error or otherwise perfect. When we discover errors or omissions, or they are brought to our attention, we endeavor to correct them in succeeding releases of the product.

Please send any constructive comments on the contents of this document to the following email address: support@garantir.io

Overview

This document details how to install, configure, operate, troubleshoot, and uninstall the GaraSign CryptoTokenKit provider and custom code and product signing utilities for macOS. The CryptoTokenKit provider allows users to use native signing tools like codesign and productsign on macOS while offloading the cryptographic processing to GaraSign. The custom utilities sign data that is compatible with the codesign and productsign utilities provided by Apple and is provided for backwards compatibility for existing customers who already use these tools. Regardless of which signing method is chosen, verification of the signed bundles can always be done by Apple's native codesign and productsign utilities.

Intended Audience

All products produced by Garantir are designed to be installed, configured, operated, and maintained by personnel with the necessary knowledge, skill, training, and qualifications to safely perform their duties. This document is intended for personnel responsible for installing, configuring, operating, and/or troubleshooting signing clients on macOS systems. It is assumed that the readers of this document and the users of its content are proficient with:

- The macOS operating system
- Working from the terminal

- Security concepts including, but not limited to, authentication, authorization, digital signatures, and logging
- Apple's codesign and productsign utilities

Document Conventions

This document uses the following conventions to easily distinguish commands and alert you to important information.

Commands

Commands to be executed from a terminal are provided in italics with a gray background, as shown below:

exampleCommand -switch1

In some cases, commands will need to be edited with customer-specific values. These parts of the commands will be highlighted and in brackets, as shown below:

exampleCommand -switch1 <insert some value>

Warnings

To help reduce the chance of data loss or corruption, this document provides warnings inside a red box with a warning icon, as shown below:

> **Warning**: Always exercise caution when performing security-related duties.

Components

This macOS signing bundle is made up of the following components:

1. GaraSign.pkg – Installs the GaraSign macOS client integration.
2. GaraSignKeychainManager.app – Contains the CryptoTokenKit plugin.
3. GaraSignKeyChainManagerReloader – Synchronizes the user's key permissions with the keychain.
4. codesignGRS – (Deprecated) Produces signatures that are compatible with Apple's codesign utility.
5. productsignGRS – (Deprecated) Produces signatures that are compatible with Apple's productsign utility.
6. GaraSignUninstaller – Uninstalls the GaraSign macOS client integration.
7. grspassword – A utility to obfuscate values in the configuration file.

Supported Mechanisms

Signature

The GaraSign macOS integration supports signing data using RSA with SHA-256. While other signature schemes are supported, the native macOS signing tools define the signature parameters.

Authentication

These utilities support the following authentication methods:

1. Client-Certificate (i.e., Mutual TLS)
2. Username and Password
3. Kerberos

Installation & Configuration

Prerequisites

The following items are needed to install this utility:

1. A computer running one of the following versions of macOS:
    a. 14 “Sonoma”
    b. 13 “Ventura”
    c. 12 “Monterey”
2. Administrator access to the target computer

Install

There are two methods to install the GaraSign macOS client integration:

1. Graphical User Interface (GUI)
2. Command Line

The GUI approach is the more user-friendly approach for end users and is designed for users performing the installation on a single computer. The command line approach is designed for system administrators who need to perform installation on many computers.

Graphical User Interface

To install via the GUI, perform the following steps:

1. Double click on the GaraSign.pkg installer
2. When prompted, enter your password
3. Wait until the installer completes

Command Line

To install via the command line, perform the following steps:

1. Open a terminal window
2. Navigate to the directory containing the GaraSign.pkg file
3. Execute the following command: *sudo installer GaraSign.pkg -target /*
4. Wait for installation to complete

Configuration

Configuration is done by editing the configuration file, located at the following two locations:

1. /etc/Garantir/codesignGRS/config.ini (global path)
2. $HOME/.config/Garantir/codesignGRS/config.ini (local path)

If the config.ini file is located under the local path, then the one under the global path is ignored. Otherwise, the config.ini from the global path is copied to the local path and then used.

Property Name	Section	Property Value
Name	Server	[String] The DNS name or IP address of GaraSign
Port	Server	[int] The port number that the GaraSign listens on
URL	Server	[String] The path in the URL of the GaraSign

Enable	Cache	[int] Set to 1 to cache the session token. Set to 0 to not cache the session token. Default = 1.
TTL	Cache	[int] The time, in seconds, that the cache value is considered valid. Default = 300.
Enable	Log	[int] Set to 1 to have log files generated. Set to 0 to not have log files generated. Default = 0.
Folder	Log	[String] The directory to write log files to.
count	users	[int] For future functionality. Leave as 1 for now.
authentication	user1	[String] The authentication method to use. Options are *Kerberos* and *Normal*. Use *Kerberos* to enable Kerberos via SPNEGO. Use *Normal* to enable certificate authentication (i.e., mutual TLS) or username and password authentication.
AllowGlobalKerberos	user1	[int] Set to 0 to restrict Kerberos authentication to intranet sites. Set to 1 to enable Kerberos authentication to all sites, including those on the internet. Default = 0.
username	user1	[String] The username to authenticate with if authenticating with username and password. This value can optionally be obfuscated by the grspassword utility.
password	user1	[String] The password to authenticate with if authenticating with username and password. This value can optionally be obfuscated by the grspassword utility.
P12File	user1	[String] The absolute path to the PKCS12 client certificate file to use for client-certificate authentication.
P12Pass	user1	[String] The password to the PKCS12 file to use for client authentication. This value can optionally be obfuscated by the grspassword utility.
KeychainIdentity	user1	[String] The identity of the certificate in the keychain to use for client certificate authentication. This is the preferred method to configure client certificate authentication.

Once the configuration file has been edited, the CA certificates for TLS certificate chain validation must be configured. Ensure that the CA certificates have been added and appropriately trusted in the keychain.

Obfuscating Sensitive Values

Users may optionally obfuscate the username, *password* and *P12Pass* values in the configuration file by using the grspassword utility. This utility provides some protection of these sensitive values but should be coupled with proper access control and security practices.

> ⚠ **Warning**: Obfuscation only applies limited security. With enough effort, an attacker can reverse the obfuscation process. Always use security in depth and couple obfuscation with access control.

To obfuscate the *password* value, perform the following steps:

1. Open a terminal window
2. Execute the following command: */Library/Garantir/bin/grspassword*
3. Type the username and then press Enter

4. Type the password and then press Enter
5. Confirm the password and then press Enter
6. Copy the obfuscated username and password values and place them in your config.ini file

To obfuscate the *P12Pass* value, perform the following steps:

1. Open a terminal window
2. Execute the following command: */Library/Garantir/bin/grspassword --p12*
3. Type the PKCS#12 file's password and then press Enter
4. Confirm the PKCS#12 file's password and then press Enter
5. Copy the obfuscated PKCS#12 password and place it in your config.ini file

Usage

List Keychain Entries

Even though the key objects are loaded into the keychain, they are not visible by the standard GUI keychain utility. To view your GaraSign keys/certificates, perform the following steps:

1. Open a terminal as your standard user
2. Execute the following command: *sc_auth identities*
3. Your keychain entries can now be referenced by the value in the parentheses of the command's output

```
garantir@garantir ~ % sc_auth identities
SmartCard: io.garantir.GaraSignKeychainManager.GaraSignExtension:GRSID-123456789
Unpaired identities:
2B48E77FF0C288D2971C2F2A88FCC992467E671B        user=0:key=0:CERT (rsa_code_signer)
C730FCEDA7827CD8BDADA67980BF15BAC8EDEB3D        user=0:key=1:CERT (Developer ID Installer: GARANTIR (Z933746L9S))
CC6284D403B57D1CE897761A472A3AD7AD1E17AD        user=0:key=2:CERT (ec_code_signer)
D73701D050D896F6AD473D4D554386CD5F177205        user=0:key=3:CERT (Developer ID Application: GARANTIR (Z933746L9S))
DE833C5276CF657DB6E6707D6B92B21A7E73B9F4        user=0:key=4:CERT (mutual_tls)
```

Application Signing

To sign an application, use the macOS codesign utility and specify the appropriate identity from your keychain.

codesign --sign "<identity>" -f </path/to/app>

Example: codesign --sign "Developer ID Application: GARANTIR (Z933746L9S)" -f unsigned.app

Installer Signing

To sign an installer, use the macOS productsign utility and specify the appropriate identity from your keychain.

productsign --sign "<identity>" </path/to/installer> </output/path>

Example: productsign --sign "Developer ID Installer: GARANTIR (Z933746L9S)" unsigned.dmg signed.dmg

Legacy Signing Methods

> **Warning**: The tools mentioned in this section are deprecated and will be removed in a future release of GaraSign. Customers new to macOS signing with GaraSign should use the native macOS signing tools. Customers currently using legacy signing tools should migrate to native tools (i.e., codesign and productsign) as soon as possible.

codesignGRS

To sign a binary with codesignGRS, execute the following command from a terminal window:

codesignGRS -s <key name> -u <username> -p <password> <file to sign>

Note: This command assumes that *codesignGRS* is configured in the *Path* variable. If it is not, the full path (i.e., /Library/Garantir/bin/codesignGRS) will need to be used.

To use client certificate authentication, set the configuration file and then use the following command:

codesignGRS -s <key name> -p <keystore password> <file to sign>

To specify a timestamping URL, add the *--timestamp* option as shown below:

codesignGRS -s <key name> -p <keystore password> <file to sign>--timestamp=<TSA URL>

Verify Signature

Verifying the signature can be done with Apple's codesign utility. Example:

codesign --verify --verbose <signed file>

productsignGRS

To sign a binary with productsignGRS, execute the following command from a terminal window:

productsigngrs --sign <key name> --in </path/to/input/pkg> --out </path/to/signed/pkg>

Note: This command assumes that *productsigngrs* is configured in the *Path* variable. If it is not, the full path (i.e., /Library/Garantir/bin/productsigngrs) will need to be used.

Verify Signature

Verifying the signature can be done with Apple's pkgutil utility. Example:

pkgutil --check-signature <signed file>.

Uninstall

To uninstall this package, perform the following steps:

1. Open a terminal window
2. Execute the following command: *sudo /Library/Garantir/bin/GaraSignUninstaller*
3. Wait for the uninstaller to complete

Frequently Asked Questions

Can I sign using a key/certificate in my macOS Keychain?

Yes, by using the native codesign and productsign tools you can. However, if you use codesignGRS or productsignGRS, you will not be able to use keychain certificates. Customers currently using codesignGRS and/or productsignGRS are strongly advised to migrate to native macOS signing tools as both codesignGRS and

productsignGRS are now deprecated and may be removed in a future release. Customers deploying new GaraSign macOS clients should avoid using codesignGRS and productsignGRS.

Does codesignGRS and/or productsignGRS need to be installed on customer computers that want to use our signed binaries?

No, these tools are only used to sign, not to verify signatures. Once signed, your binaries can be distributed to your customers and they will verify on their macOS systems that do not have the GaraSign tools installed on them. Additionally, these tools are deprecated and should no longer be used by customers. Please migrate to native macOS signing tools as soon as possible.

Am I still satisfying my requirement to use an HSM?

The short answer is yes. The GaraSign server that these tools communicate with sits in front of your company's cryptographic tokens. In most cases, these tokens are HSMs although GaraSign supports devices other than HSMs as well. Contact your GaraSign administrator to find out more about your organization's architecture and cryptographic tokens.

Troubleshooting

In general, most problems can be diagnosed by enabling logging and viewing the log file. Provided below is a list of common error messages one might see while signing and some suggested ways to resolve each issue.

No Keychain Items Displayed

Description: No items displayed after executing *sc_auth identities*.

Cause 1: There is an authentication issue.

Resolution 1: Either a network connection or invalid authentication credentials is causing the GaraSign provider to not be able to complete its requests to the GaraSign server. Ensure network connectivity and make sure your authentication credentials are correct.

Cause 2: Cache has become corrupted.

Resolution 2: Perform the following steps:

1. Delete the contents of the $HOME/.cache/GRS folder
2. From a terminal window, execute the following command: */Library/Garantir/bin/GaraSignKeychainManagerReloader*
3. Rerun *sc_auth identities*

Key Permission Error

Description: Receive an error stating that the user does not have permissions to the signing key even though the user does have permission to the signing key.

Cause: Cached session token does not map to the updated permissions.

Resolution: Either disable caching or delete the contents of the $HOME/.cache/GRS folder and try again.

Appendix A

Glossary of Terms, Abbreviations, and Acronyms

Certificate Authority (CA)	An entity that issues digital certificates.
Elliptic-Curve Diffie-Hellman (ECDH)	A variant of the Diffie-Hellman protocol that uses elliptic-curve cryptography.
Elliptic-Curve Digital Signature Algorithm (ECDSA)	A variant of the Digital Signature Algorithm which uses elliptic-curve cryptography.
Hardware Security Module (HSM)	A physical computing device that safeguards, manages, and makes use of cryptographic keys.
INI	An informal configuration file format.
Message Digest 5 (MD5)	A legacy cryptographic hashing algorithm.
Optimal Asymmetric Encryption Padding (OAEP)	A padding scheme standardized in PKCS#1 v2.
Public Key Cryptography Standards (PKCS)	A group of public-key cryptography standards published by RSA Security LLC.
Public Key Infrastructure (PKI)	A set of roles, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates and manage public-key encryption.
Registration Authority (RA)	The authority in a PKI that verifies requests for a digital certificate.
Rivest-Shamir-Adleman (RSA)	One of the first public-key cryptosystems.
Secure Hash Algorithm (SHA)	A set of cryptographic hashing with various output lengths.
Transport Layer Security (TLS)	A cryptographic protocol designed to provide communications security over a computer network.