Enterprise Code Signing Policy

Updated May 22, 2025

1. Overview

Multiple groups within modern organizations publish software for internal and external use. To help protect against software tampering and confirm the authenticity of software releases, best practices state that software binaries should be signed with cryptographic keys. However, trust in these signatures requires appropriate controls and monitoring of code signing processes.

This policy is intended to help any organization that creates and distributes software, even though calling it an "enterprise" policy may connote a large organization.

2. Purpose

This policy outlines the requirements and procedures for controlling software code signing processes to ensure that only authorized, verified code is signed and distributed, enhancing the security and integrity of the software.

This policy will ensure the organization knows what it is publishing, and that the published software does not introduce unacceptable security risks. It will also allow users to verify the integrity of published files to demonstrate that they have not been tampered with and alert users if software was not created by the publisher attached to the code.

A rigorous code signing policy with good record keeping can also facilitate the work of auditors, both internal and external.

3. Scope

This policy applies to all personnel, processes, and systems involved in the code signing process within the organization, including internal and external developers, signers, reviewers, and auditors. It applies to code that is developed by the organization for internal use or external sale, and software artifacts created or used in the SDLC and CI/CD process (e.g., SBOMs, Build Scripts).

It does not apply to third-party software the organization uses for development or other activities. (e.g., productivity tools like Microsoft Office or development tools like JFrog).

4. Access and Privileges

4.1 Authentication

Users accessing the software development environment must use approved authentication services or methods. They must not use shared accounts, insecurely stored credentials, local accounts, or other means of authentication that violate the enterprise authentication policy.

Multi-factor authentication or server-to-server authentication must be used to access and authorize signing per CA/B Forum baseline requirements.

4.2 Define Roles – Role Based Access Control (RBAC)

Under best practices, the organization should establish separate roles with defined privileges for code signing activities. These roles should minimally include Submitter, Signer, and Security Officer and expand to include Reviewer, Auditor, and System Administrator.

4.3 Separation of Duty

Users may have multiple roles based on commercially reasonable practices. However, no single user may have sole control over a process. This introduces a single point of failure for security.

Limit access to the minimum set of privileges to have tighter security control per the Least Privilege Security Principle.

4.4 Limit Access to Core Resources

All access, but especially access to privileged resources, like build servers and HSM administration, should be assigned, based on the defined roles and not specific users (or unrelated groups like Administrator or C-Suite).

The minimum set of access and privileges should be given to core resources to have tighter control over security.

4.5 High Risk Actions

Certain operations, such as keypair export, keypair deletion, and certificate revocation, can be high security risks or cause major disruptions in processes if not done correctly. These types of actions may justify requiring more than one authorization for signing, in addition to heavily restricted signing privileges.

4.6 Risk of Compromised Software to Cause Damage

The risk of each software product or project must be assessed based on the potential damage if the software contained malicious code or exploitable vulnerabilities that would result in unauthorized access to data, loss of system functionality, or other failure (e.g., software with low-level system access, such as root access in Linux).

Sufficiently high-risk software may justify requiring more than one authorization for signing in addition to heavily restricted signing privileges.

5. Certificate and Keypair Management

5.1 Software Environments

Unique code signing keypairs and certificates must be used for non-production and production environments. Test environments that are remotely accessible, and process sensitive data, must be treated as production environments. privileges.

5.2 Certificate Inventory Keypair Storage

Production code signing keys must be generated and stored securely, using a hardware security module compliant with FIPS 140-2 Level 2 or Common Criteria EAL 4+ and used according to enterprise Cryptographic Management and Certificate policies.

Multiple users must not share production code signing keys unless activity logging can record and identify actions by unique users, including service users.

5.3 Certificate Inventory and Lifecycle

Maintain an inventory of all code signing certificates. Monitor expiration dates and generate new certificates proactively.

New certificates and keys can be created securely and put into use automatically through most Certificate Lifecycle Management (CLM) systems.

5.4 Key Rotation

Rotating keys improves security by reducing the impact of a compromised key. Key rotation can be done manually, but it is easier and faster to automate the process.

5.5 Key Encryption

Encrypt keys at the highest level possible based on the decryption methods the end user can accommodate. Encourage the ‘other end’ to upgrade their decryption options, so that PQC-ready encryption algorithms can be used when they become available.

5.6 Public vs. Private PKI

Software that is used or accessed outside your organization should be signed using a public key that can easily be verified through standard PKI lists.

Software that will only be used and accessed through a corporate network can use “Private PKI,” where certificates are chained back to a trusted root but are issued by the company through an Intermediary Certificate Authority (ICA). This is a common situation for software that is developed and used only internally.

6. Code Signing Process

6.1 Security Reviews

Software source code must pass a security review according to the Software Development Life Cycle (SDLC) security policy.

Each release and associated components must be scanned for malicious indicators and known vulnerabilities. Do not sign software if it contains confirmed malware or an exploitable vulnerability.

6.2 Automated Processes

Best practice is to automate the code signing process as part of the SDLC and CI/CD pipeline. Use an approved Certificate Authority (CA) for code signing certificates. Ideally, developers should use a unique signing key to sign any code they check-in. The individual signatures must be verified before signing a release.

6.3 Timestamps

Releases that will be valid after the code signing certificate has expired must include a timestamp from an approved timestamping service to show proof of signing time. Be sure that your timestamps and Timestamp Authority (TSA) meet the new CA/B Forum security requirements enforced starting April 2025.

6.4 Signing Release Artifacts and Software Bills of Materials (SBOMs)

Release artifacts, including the software code branch, SBOM, build scripts, and Vulnerability Exchange Documents, must be signed and stored to help detect tampering with assertions and archives.

Artifacts should be stored securely and may need methods for sharing them with customers or governing bodies. SBOMs, in particular, may be part of a quote package or product submission.

6.5 Auditing and Logs

Maintain a comprehensive audit trail including proof of code reviews, security scans, and code signing key action. Activities must be associated with unique users and service accounts. Signing, and any actions associated with certificates, keypairs, users, and templates should be captured in the logs.

7. Crypto-Agility – Preparation and Response to Change

7.1 Preparation and Response to Incidents and Unexpected Change

Define and automate as many procedures as possible for quickly documenting, revoking, and reissuing compromised keys.

Define and automate procedures for re-signing and re-deploying code and software artifacts that were signed with the compromised key.

7.2 Preparation and Response to Expected Changes

Regulations are constantly changing. However, they often are phased in or provide time to make system changes. Define and automate procedures for documenting and making changes progressively, for instance by making updates when reissuing an expiring certificate.

8. Review and Updates

This policy will be reviewed annually and updated as necessary to ensure its effectiveness and alignment with industry standards and organizational changes.

Download a word document of this file for your personal use by clicking **here**.