OpenChain Security Assurance

Reference Specification

Version 2.0

*OpenChain Project: Building Trust In The Supply Chain Since 2016*



Contents

[Introduction iii](#_Toc79496715)

[1 Scope 1](#_Toc79496716)

[2 Terms and definitions 1](#_Toc79496717)

[3 Requirements 2](#_Toc79496718)

[3.1 Program foundation 2](#_Toc79496719)

[3.1.1 Policy 2](#_Toc79496720)

[3.1.2 Competence 2](#_Toc79496721)

[3.1.3 Awareness 2](#_Toc79496722)

[3.1.4 Program scope 3](#_Toc79496723)

[3.1.5 Standard Practice Implementation 3](#_Toc79496724)

[3.2 Relevant tasks defined and supported 3](#_Toc79496725)

[3.2.1 Access 3](#_Toc79496726)

[3.2.2 Effectively resourced 4](#_Toc79496727)

[3.3 Open Source Software content review and approval 4](#_Toc79496728)

[3.3.1 Bill of materials 4](#_Toc79496729)

[3.3.2 Security Assurance 5](#_Toc79496730)

[3.4 Adherence to the guideline requirements 5](#_Toc79496731)

[3.4.1 Completeness 5](#_Toc79496732)

[3.4.2 Duration 5](#_Toc79496733)

Introduction

The OpenChain Project is working towards a supply chain where open source is delivered with trusted and consistent compliance information. We maintain OpenChain ISO/IEC 5230:2020, the International Standard for open source license compliance. Adjacent to this the project maintains a large international community, extensive reference materials, and working groups addressing various domain issues. We support discussions around security, export control, M&A and other topics.

OpenChain ISO/IEC 5230:2020 is a process management specification that identifies inbound, internal and outbound inflection points where a process, policy or training should exist. The identification and tracking of software used and deployed is an inherent part of getting this right, and this also allows our standard to also be useful for security or export control.

We noticed that OpenChain ISO/IEC 5230:2020 was being used quite often in deployment discussions and we wanted to support our broader community around these use-cases. The reference specification you are now reading is focused on the security domain. It is intended to identify and describe the key requirements of a quality Security Assurance Program in the context of using Open Source Software. This early iteration of the document focuses on a narrow subset of primary concern: checking Open Source Software against publicly known security vulnerabilities like CVEs, GitHub/GitLab vulnerability reports, and so on.

This document focused on the “what” and “why” aspects of a quality Security Assurance Program rather than delving into to “how” and “when.” This is a conscious decision to ensure flexibility for companies of any size and in any market to use this reference specification. This approach, along with the types of processes identified, is built on more than half a decade of practical global feedback around the creation and management of such programs. The end result is that a company can frame a program that precisely fits their supply chain requirements, scoped to a single product or a complete legal entity, and take this solution to market quickly and effectively.

The scope of this reference specification may expand over time based on community feedback.

This introduction describes the reference specification’s purpose. Section 2 defines key terms used throughout this document. Section 3 defines the requirements that a Program must satisfy to achieve a core level of Security Assurance. Each requirement consists of one or more verification materials (i.e., records) that must be produced to satisfy the requirement. Verification materials are not required to be made public, though an organization may choose to provide them to others, potentially under a Non-Disclosure Agreement (NDA).

This reference specification is licensed under [Creative Commons Attribution License 4.0](https://creativecommons.org/licenses/by/4.0/) (CC-BY-4.0). Because it takes the form of a Reference Specification, and is therefore intended to fit into the mental model applied to specification creation, it is not designed to be modified outside of the formal editing track. You can take part in editing this document via the OpenChain Project bi-weekly calls. You can learn about joining these calls and our other activities here:

https://www.openchainproject.org/community

OpenChain Security Assurance Reference Specification

# Scope

This document specifies the key requirements of a quality Open Source Software Security Assurance Program that establishes trust between organizations exchanging software solutions comprised of Open Source Software.

# Terms, Definitions and Examples

For the purposes of this document, the following terms and definitions apply.

**2.1 CVE**

Common Vulnerabilities and Exposures (CVE) is a public database of disclosed computer software security issues and flaws. When someone refers to a CVE, they mean a security flaw that's been assigned a CVE ID number within the database. The CVE database is sponsored by the US Department of Homeland Security (DHS) and the Cybersecurity and Infrastructure Security Agency (CISA).

**2.2** **Known Vulnerability (Vulnerabilities)**

Security vulnerabilities that were discovered in Open Source Software components that are publicly available. That would include any publicly published vulnerabilities including but not limited to CVEs, GitHub/GitLab vulnerability alerts, package manager alerts and so forth.

**2.3 Open Source Software**

software subject to one or more licenses that meet the Open Source Software Definition published by the Open Source Software Initiative (see www.opensource.org/osd) or the Free Software Definition published by the Free Software Foundation (see www.gnu.org/philosophy/free-sw.html) or similar license

**2.4 Program**the set of policies, processes and personnel that comprise an organization’s security assurance activities

**2.5 Program Participants**any organization employee or contractor that defines, contributes to or has responsibility for preparing Supplied Software. Note: Depending on the organization, that may include (but is not limited to) software developers, release engineers, quality engineers, product marketing and product management.

**2.6 Security Assurance**the confidence that a system meets the requirements for security best practices and is resilient against Known Vulnerabilities.

**2.7 SPDX**the format standard created by the Linux Foundation’s SPDX (Software Package Data Exchange) Working Group for exchanging bill of materials information for a given software package, including associated license, copyright information and Known Vulnerabilities (see [spdx.org](https://spdx.org/))

**2.8 Supplied Software**software that an organization distributes or makes available to third parties (e.g., other organizations or individuals)

**2.9 Verification Materials**materials that demonstrate that a given requirement of the reference specification is satisfied

# Requirements

## Program foundation

### Policy

A written policy shall exist that governs Open Source Software Security Assurance of the Supplied Software. The policy shall be internally communicated.

**Verification material(s):**

* 3.1.1.1 A documented Open Source Software Security Assurance policy.
* 3.1.1.2 A documented procedure that makes Program Participants aware of the existence of the Security Assurance policy (e.g., via training, internal wiki, or other practical communication method).

**Rationale:**

To ensure steps are taken to create, record and make Program Participants aware of the existence of an Open Source Software Security Assurance policy. Although no requirements are provided here on what should be included in the policy, other sections may impose additional requirements.

### Competence

The organization shall

* Identify the roles and the corresponding responsibilities of those roles that affects the performance and effectiveness of the Program;
* Determine the necessary competence of Program Participants fulfilling each role
* Ensure that Program Participants are competent on the basis of appropriate education, training, and/or experience;
* Where applicable, take actions to acquire the necessary competence; and
* Retain appropriate documented information as evidence of competence.

**Verification material(s):**

* 3.1.2.1 A documented list of roles with corresponding responsibilities for the different Program Participants.
* 3.1.2.2 A document that identifies the competencies for each role.
* 3.1.2.3 List of participants and their roles
* 3.1.2.4 Documented evidence of assessed competence for each Program Participant.

**Rationale:**

Ensure that the Program Participants have obtained a sufficient level of competence for their respective roles and responsibilities.

### Awareness

The organization shall ensure that the Program Participants are aware of:

* The Open Source Software Security Assurance policy;
* Relevant Program objectives;
* Their contribution to the effectiveness of the Program; and
* The implications of not following the Program’s requirements.

**Verification material(s):**

* 3.1.3.1 Documented evidence of assessed awareness for the Program Participants - which should include the Program’s objectives, one’s contribution within the Program, and implications of Program non-conformance.

**Rationale:**

To ensure the Program Participants have obtained a sufficient level of awareness for their respective roles and responsibilities within the Program.

### Program scope

Different Programs may be governed by different levels of scope. For example, a Program could govern a single product line, an entire department or an entire organization. The scope designation needs to be declared for each Program.

**Verification material(s):**

* 3.1.4.1 A written statement that clearly defines the scope and limits of the Program.
* 3.1.4.2 A set of metrics the program shall achieve to improve

**Rationale:**

To provide the flexibility to construct a Program that best fits the scope of an organization’s needs. Some organizations could choose to maintain a Program for a specific product line while others could implement a Program to govern the Supplied Software of the entire organization.

### Standard Practice Implementation

The Organization demonstrates a sound and robust handling procedures of Known Vulnerabilities and Secure Software Development by defining and implementing following procedures:

* Method to identify structural and technical threats to the Supplied Software is defined
* Method for detecting existence of Known Vulnerabilities in Supplied Software
* Method for following up on identified Known Vulnerabilities
* Method to communicate identified Known Vulnerabilities to customer base when warranted
* Method for analyzing Supplied Software for newly published Known Vulnerabilities post release
* Method for continuous and repeated Security Testing is applied for all Supplied Software before release
* Method to verify that identified risks will have been addressed before release of Supplied Software

A process shall exist for the Security Assurance methods listed above.

**Verification material(s):**

* 3.1.5.1 A documented procedure exists for each of the methods identified above.

**Rationale:**

To ensure appropriate processes exists for detecting and following up on Known Vulnerabilities in the Supplied Software.

## Relevant tasks defined and supported

### Access

Maintain a process to effectively respond to Known Vulnerability external inquiries. Publicly identify a means by which a third party can inquire about a Known Vulnerability with respect to a given software offering.

**Verification material(s):**

* 3.2.1.1 Publicly visible method that allows any third party to make a Known Vulnerability inquiry (e.g., via a published contact email address – security@company.com, opensource@company.com, …).
* 3.2.1.2 An internal documented procedure for responding to third party Known Vulnerability inquiries exists

**Rationale:**

To ensure there is a reasonable way for third parties to contact securely the organization regarding security vulnerability inquiries and that the organization is prepared to respond.

### Effectively resourced

Identify and Resource Program Task(s):

* Assign accountability to ensure the successful execution of Program tasks.
* Program tasks are sufficiently resourced:
  + Sufficient time to perform the tasks have been allocated; and
  + Adequate funding has been allocated.
* A process exists for reviewing and updating the policy and supporting tasks; and
* Technical expertise pertaining to Known Vulnerabilities is accessible to those who may need such guidance.

**Verification material(s):**

* 3.2.2.1 Document with name of persons, group or function in Program role(s) identified.
* 3.2.2.2 The identified Program roles have been properly staffed and adequate funding provided.
* 3.2.2.3 Identification of expertise available to address identified Known Vulnerabilities.
* 3.2.2.4 A documented procedure that assigns internal responsibilities for Security Assurance.
* 3.2.2.5 A documented procedure for handling the review and remediation of identified Known Vulnerability cases.

**Rationale:**

To ensure: i) Program responsibilities are effectively supported and resourced and ii) policies and supporting processes are regularly updated to accommodate changes in Security Assurance best practices.

## Open Source Software content review and approval

### Software Bill of Material (SBOM)

A process shall exist for creating and maintaining a bill of materials that includes each Open Source Software component from which the Supplied Software is comprised.

**Verification material(s):**

* 3.3.1.1 A documented procedure for identifying, tracking, reviewing, approving, and archiving information about the collection of Open Source Software components from which the Supplied Software is comprised.
* 3.3.1.2 Open Source Software component records for the Supplied Software that demonstrates the documented procedure was properly followed.

**Rationale:**

To ensure a process exists for creating and managing an Open Source Software component bill of materials used to construct the Supplied Software. A bill of materials is needed to support the systematic review of each component to understand if any Known Vulnerabilities exist

### Security Assurance

* For each Open Source Software component in the bill of materials for the Supplied Software release under review
  + Apply method for detecting existence of Known Vulnerabilities
  + For each identified Known Vulnerability assign a risk/impact score
  + Depending on the risk/impact score take the appropriate action (e.g., contact customers if warranted, upgrade component, no further action, …)
  + If Known Vulnerability is present in previously distributed Supplied Software, depending on the risk/impact score take the appropriate action (e.g., contact customers if warranted)
* Post software solution release - ability to later identify newly reported Known Vulnerabilities that may impact a Supplied Software solution and to respond accordingly

**Verification material(s):**

* 3.3.2.1 A documented procedure for handling detection and resolution of Known Vulnerabilities for the Open Source Software components of the Supplied Software.
* 3.3.2.2 For each Open Source Software component a record is maintained of the identified Known Vulnerabilities and action(s) taken (including even if no action was required)

**Rationale:**

To ensure the Program is sufficiently robust to handle the identified Known Vulnerabilities for the Open Source Software from which the Supplied Software is comprised. That a procedure exists to support this activity and that the procedure is followed.

## Adherence to the guideline requirements

### Completeness

For a Program to be deemed conformant with this reference specification, the organization shall affirm that the Program satisfies the requirements presented in this document.

**Verification material(s):**

* 3.4.1.1 A document affirming the Program specified in §3.1.4 satisfies all the requirements of this document.

**Rationale:**

To ensure that if an organization declares that it has a Program that is conforming, that the Program has met all the requirements of this document. The mere meeting of a subset of these requirements is not considered sufficient.

### Duration

A Program that is conformant with this version of the reference specification shall last 18 months from the date conformance validation was obtained.

**Verification material(s):**

* 3.4.2.1 A document affirming the Program meets all the requirements of this reference specification, within the past 18 months of obtaining conformance validation.

**Rationale:**

It is important for a Program to remain current with the reference specification requirements if an organization wants to assert conformance over time. This requirement ensures that the Program’s supporting processes and controls do not erode if an organization continues to assert Program conformance over time.