RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

BACK TO BASICS

HOW TO SELECT AN OPEN-SOURCE SOFTWARE

Choosing any product (tool, library, framework) requires careful thought for any organisation, as it can have a major impact on the security of the information system. In the case of an open-source product, the criteria may differ from those applicable to proprietary products.

This document presents ten key points to help entities become aware of cybersecurity issues and make informed choices. Note that it is not necessary for a project to achieve a perfect score.

→ **Review the project history and reputation.** Low activity may indicate a high level of maturity and stability, but it may also reflect a lack of resources to support the project and develop new features. The following factors may be taken into account:

> date of creation, regularity of contributions, number and experience of main contributors;

> the maturity of the project, as estimated by its developers (e.g.: beta version) or by the ecosystem (users, trade press, SILL, etc.).

→ **Evaluate the software's maintenance in operational condition (MOC)** based on:

> the declaration of designated maintainer(s);

> commits, recent versions (over the last year) and corrections of functional bugs;

> changes within the project ecosystem (underlying platforms, libraries used, etc.), which often require adaptation or porting efforts.

→ **Evaluate the maintenance of security conditions (MCS),** including:

> the presence of a point of contact for security issues, as well as a public vulnerability management procedure;

> the number of security patches and the time taken to correct critical vulnerabilities. Although it is normal over the lifespan of a project to be faced with vulnerabilities, it is crucial for the project to be able to receive and deal with this type of bug report;

> see the guide to implementing a coordinated vulnerability disclosure process for open-source projects edited by the Open Source Security Foundation (OSSF).

→ **Inventory and monitor project dependencies** (shared libraries, frameworks, etc.) by:

> checking for the availability of a dependency list (software bill of materials (SBOM) in a format such as SPDX or CycloneDX);

> ensuring that dependencies are up to date and free of known vulnerabilities;

> using dependency management tools which include information on vulnerabilities.

→ **Identify any third-party security analyses**, such as the declaration of a security visa issued by ANSSI, the French Cybersecurity Agency, or evaluation reports from the open-source community.

V1.0 (05/25)

www.cyber.gouv.fr / conseil.technique@ssi.gouv.fr

- **Assess the quality of the technical base,** as evidenced by:
  - user and administrator documentation;
  - default configuration and recommendations for secure deployment;
  - use of open, proven standards and protocols.
- **Identify any declarations of development practices:**
  - compliance with secure implementation recommendations. Refer to ANSSI's Rules for secure C language software development and Programming rules to develop secure applications with Rust, the OWASP Developer Guide, and NSA's Case for memory safe roadmaps;
  - peer code review and tool-based code analysis;
  - unit testing and continuous integration;
  - reproducibility of generated binaries;
  - strong security properties of the languages used (memory management security, strong typing, etc.).
- **Conduct regular audits on the evolution of the project contributions and on its security**, combining manual and tool-based analysis. The aim is to protect against the insertion of illegitimate code, as illustrated by the case of the XZ-utils project: a malicious contributor had inserted a backdoor enabling the takeover of the workstations on which this library was installed.
- **Subscribe, if possible, to a support contract (functional or security) with a community or with a commercial entity**. Such a contract can help support the project by providing financial stability and sustainability, while reassuring the entity deploying it.
- **Verify whether the project or its maintainers have expressed a need for support** (financial or in terms of contributors, maintenance, code, security, etc.). These explicit calls testify to the state of health of the project, and can also be used to help it and ensure its continuity.