

C/C++ SBOM Tool Evaluation Checklist

Not all SBOM tools are built for embedded C/C++. Use this checklist to evaluate whether a solution can accurately capture what's in your software, connect vulnerabilities to the right products, and support your security and compliance needs.

SBOM Generation & Coverage

☐ Build-Time Visibility

Does the tool capture what was actually compiled into the product?

☐ C/C++ Build Support

Does it work with your build systems, such as CMake, Make, Ninja, Bazel, or Yocto?

☐ Static Dependencies

Can it identify libraries statically linked into the final binary?

☐ Vendored Code

Can it detect third-party code copied directly into your source tree?

☐ Conditional Compilation

Does the SBOM reflect the components included in each specific build configuration?

Accuracy & Actionability

☐ Component

Can you validate that expected components are present and identify what may be missing?

☐ Version Precision

Does the tool reliably identify exact component versions rather than returning "unknown" or estimates?

☐ Standardized Output

Does it support formats and identifiers such as SPDX, CycloneDX, purl, and CPE?

☐ Vulnerability & VEX Support

Can components be connected to vulnerabilities and product-specific VEX information?

☐ License Visibility

Does it identify open-source licenses and associated obligations?

Operational Fit

❑ Automated & Repeatable

Can SBOM generation be integrated into your build or release process with minimal manual effort?

❑ Artifact Traceability

Can every SBOM be tied back to the exact product version and released artifact?

Before You Choose

Run each tool against the same representative C/C++ build. Compare what components and versions are identified, what is missed, and how well each SBOM reflects what was actually shipped.

The key question: *Can you confidently determine what's in the product, whether it's affected, and prove it?*

