interlynk

The Enterprise SBOM Management Checklist

A practical controls checklist for managing SBOMs at scale, built around the five places enterprise SBOM management breaks and the controls that keep each one in check.

Generating SBOMs is the easy part. Managing thousands of them across hundreds of teams, suppliers, and acquisitions is where programs fall apart. Work through this as a maturity check. **If you cannot tick an item, that is a gap worth closing before the next zero-day finds it for you.**

1 Control SBOM sprawl (volume)

The folder of files and the spreadsheet stop working long before they throw an error. Decide the storage model before you hit the wall.

- [ ] SBOMs are pushed into a system of record as a build-pipeline step, not uploaded by hand.
- [ ] Every SBOM is keyed to an immutable build or release identifier (such as the artifact digest) so the right one is retrievable years later.
- [ ] The store is indexed by component, so a lookup is a query rather than a scan across millions of files.
- [ ] A retention policy keeps SBOMs for shipped and released artifacts to meet regulatory retention, and prunes throwaway CI builds.
- [ ] You can state, roughly, how many SBOMs you hold and where they live.

2 Make every SBOM agree (format and identity)

You cannot aggregate files that disagree on structure or deduplicate components that disagree on identity. Conform everything before you store it.

- [ ] Incoming SBOMs are converted to one canonical internal format and version on ingest, with the original file kept intact for audit.
- [ ] Generators are configured to emit purl as the primary component identifier, with CPE as a fallback.
- [ ] Components that arrive as free text only are flagged rather than silently accepted.
- [ ] A resolver maps aliases, package coordinates, and CPEs to a single canonical key, backed by an alias table you maintain.
- [ ] Every SBOM is validated against the spec schema at ingest, and malformed files are rejected at the door.
- [ ] Acquired companies' toolchains are routed through the same ingest and normalization path.

3 Keep SBOMs live (staleness)

A stored SBOM is a snapshot that decays. Monitoring is what turns it back into a current answer.

- [ ] Generation and evaluation are decoupled: SBOMs are generated at build time and re-evaluated continuously against fresh vulnerability intelligence.
- [ ] A new CVE resolves automatically to the exact products and versions affected.
- [ ] Findings route to the team that owns each affected product, without manual triage.
- [ ] VEX is applied so re-evaluation produces signal, and VEX statements persist so a suppression made once survives later re-scans.
- [ ] Monitoring reflects what is actually deployed or in market, not only what was built.

4 Hold suppliers to a standard (third-party SBOMs)

The third-party inventory hides the most dangerous unknowns and is usually the least managed. Set the quality bar in the contract, while you still have leverage.

- [ ] SBOM delivery is a contractual obligation that specifies format, version, and a minimum set of required elements.
- [ ] Every supplier SBOM is validated on receipt and scored for completeness and identifier quality.
- [ ] SBOMs that fall short are flagged or rejected before they enter the system of record.
- [ ] Supplier SBOMs are normalized into the same store and the same continuous monitoring as your own output.
- [ ] A named owner (typically procurement paired with security) is responsible for chasing incomplete or missing supplier SBOMs.

5 Give the program an owner (governance)

Every technical failure above has an organizational one underneath it. Diffused ownership is why the cross-portfolio question goes unanswered.

- [ ] One accountable owner exists for the program, typically a supply-chain or product security lead.
- [ ] A RACI makes the split explicit: engineering generates, a central security or platform function operates the system of record, compliance consumes.
- [ ] Policy is set centrally (format, identifiers, retention, supplier requirements) and implemented locally by each team.
- [ ] Mean time to answer “where are we exposed” is tracked as a metric.
- [ ] The zero-day question is run as a scheduled drill, before an attacker runs it for you.

What good looks like

When these controls are in place, five capabilities work together as one system. Every SBOM, internal and supplier, lives in a single system of record, whatever its format or version. Component identity is normalized, so a portfolio-wide question returns one answer rather than a reconciliation project. Stored SBOMs are re-evaluated continuously, and a newly disclosed CVE surfaces every affected product on its own. Supplier SBOMs clear the same bar as your own before they are accepted. The inventory feeds the security stack, driving vulnerability management, VEX, and incident response.

An enterprise can hold a million SBOMs and manage none of them. The paragraph above is what managing them actually looks like.
