THE BACKDOOR YOUR SBOM WILL NEVER SHOW YOU

Your SBOM is a staff list. It tells you who is in the building. It does not tell you what they did while they were there.

praedictio.ai · September 2026

You can have the best programmers in the world, but if one of them is compromised, by a third party or a nation state, your staff list will not show the backdoor they put in your code. Right names, valid credentials, working badge. What changed is what gets committed, and the change you cannot see is the one that burns you.

> **The same is true of your software.**

A software bill of materials is generated from the manifest: the packages we declared, their versions, their known vulnerabilities. That is useful, which is why it is now a procurement requirement. It is also incomplete, in a specific way.

A manifest describes what was declared at build time. It does not describe what loaded at runtime, and it says nothing about behavior. A component can sit in your SBOM with a clean bill of health and still open a socket, read a credential file, or spawn a process nobody expected. The list was accurate. The list was also silent about the thing that mattered.

A PBI SBOM™ is built the other way round. It's dynamic.

As **PBI Explainer™** watches your application run through the tests you already have, it records what loads and what each component does at runtime: which functions it calls, what it touches on the network and file system. Every component gets a checksum. The result is not a declaration. It is a record of execution.

Three things follow.

It answers the harder question. A manifest tells you whether a library is in your software. A **PBI SBOM** tells you what that library did when your software

ran: the functions it called, the files it touched, the connections it opened. For an auditor those are different kinds of evidence. One is a declaration, the other is a record.

It makes tampering visible. Every component is checksummed, so the same file at the same path returning a different checksum means something changed that nobody approved. Behavior works the same way: a component that starts doing what it never did in testing has changed, whether anyone meant it to or not.

The evidence and the control become the same artifact. The model that produced your SBOM is the model **PBI Enforcement™** applies in production. The usual pattern is to document one thing, enforce another, and hope the two do not drift apart over a year of releases. With **PBI Enforcement** they cannot. What you showed the regulator is what the application is held to.

> **TWO HONEST LIMITS.**
>
> **It does not replace your existing SBOM.** A manifest catalogues what you declared, including paths your tests never reach. PBI describes what ran.
>
> **And it is only as complete as your test coverage.** Where coverage is thin the picture is thin. It names the gap rather than papering over it, but the limit is worth stating.

Static analysis tells you what code is present. Behavior tells you what your software does. The industry has spent a decade getting very good at the first question, on the assumption that it was a proxy for the second.

It never was.

THE PBI SBOM COMES WITH PBI EXPLAINER.

Same run: the model gets built, and the SBOM falls out of it.

If you would rather see what your own software does than take our word for it:

PRAEDICTIO.AI/TRY