CleanStart

VEX and the Shift From Vulnerability Presence to Exploitability

Shifting from CVE volume to exploitability context and verifiable risk.

v2026.01.16

Why this Matters

Modern software teams are overwhelmed by vulnerabilities. Container images routinely surface hundreds of CVEs, many of which cannot be exploited in the context of the application. Security teams chase alerts. Developers triage findings they did not introduce. Remediation cycles slow releases without materially reducing risk.

This problem has intensified with the adoption of SBOMs. As visibility improves, vulnerability counts rise sharply. Teams see more issues, but gain little clarity on which ones actually matter.

Industry data consistently shows that most reported vulnerabilities do not represent real risk. Large-scale analyses indicate that **less than 10 percent of detected CVEs are typically exploitable in a given runtime context** , while teams spend the majority of their effort triaging the remainder. As SBOM adoption expands visibility, this imbalance worsens.

This is not a failure of scanners or SBOMs. It is an architectural failure in how vulnerability information is interpreted and communicated.

The Core Problem: Treating Vulnerability Presence as Risk

Vulnerability scanners do one thing well. They detect the presence of known vulnerable components. SBOMs do one thing well. They enumerate what is included in an artifact.

Neither is designed to determine exploitability.

In practice, however, organizations treat scanner output as a risk signal. Every detected CVE becomes an alert. Every alert demands triage. Every triage requires human judgment to determine whether the vulnerability is reachable, exploitable, or relevant in the deployed context.

This creates structural failure.

Alert volume without prioritization

Most vulnerabilities detected in container images are not exploitable. They may exist in unused libraries, disabled code paths, or components that are never invoked at runtime.

Scanners surface them uniformly. A critical CVE in an unreachable library appears indistinguishable from a critical CVE in an exposed service. Teams are forced to treat both as equally urgent until proven otherwise.

Manual context does not scale

Security teams often determine exploitability correctly, but manually. That context lives in tickets, emails, internal documentation, or tribal knowledge.

It does not travel with the artifact. As images are rebuilt, promoted, or shared with customers, the same vulnerabilities are re-evaluated repeatedly. The decision is correct each time, but never durable.

SBOMs amplify noise without interpretation

SBOMs increase transparency, but without exploitability context they also increase noise. Consumers receive long lists of vulnerabilities with no indication of relevance. Approval processes slow. Trust in security signals erodes.

The Architectural Mistake

The underlying mistake is architectural.

Vulnerability management systems treat **component presence** as the primary signal. Security decisions require **exploitability context** , but that context exists outside the artifact lifecycle.

Scanners identify what exists. Humans decide what matters. The decision is never preserved.

As environments scale, this model collapses.

Exploitability as an Architectural Property

To scale vulnerability management, exploitability must become a **first-class property of the software artifact** , not an after-the-fact interpretation.

This requires two conditions.

Context must be machine-readable

Human explanations do not scale across builds, deployments, or supply chains. Exploitability decisions must be expressed in a standardized, machine-readable form that tools can consume consistently.

Context must travel with the artifact

Exploitability decisions must be attached to the image or package itself. When an artifact moves through environments or to downstream consumers, its vulnerability context must move with it.

VEX as the Missing Architectural Layer

A Vulnerability Exploitability eXchange (VEX) document allows producers to communicate exploitability context for known vulnerabilities in a standardized way.

Rather than suppressing vulnerabilities, VEX adds meaning to them.

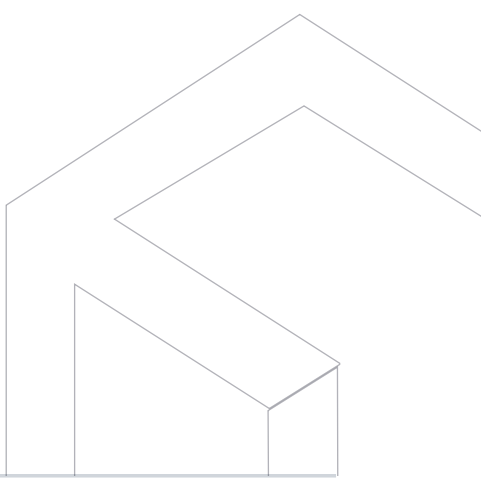
At a high level, VEX allows producers to state whether a vulnerability is:

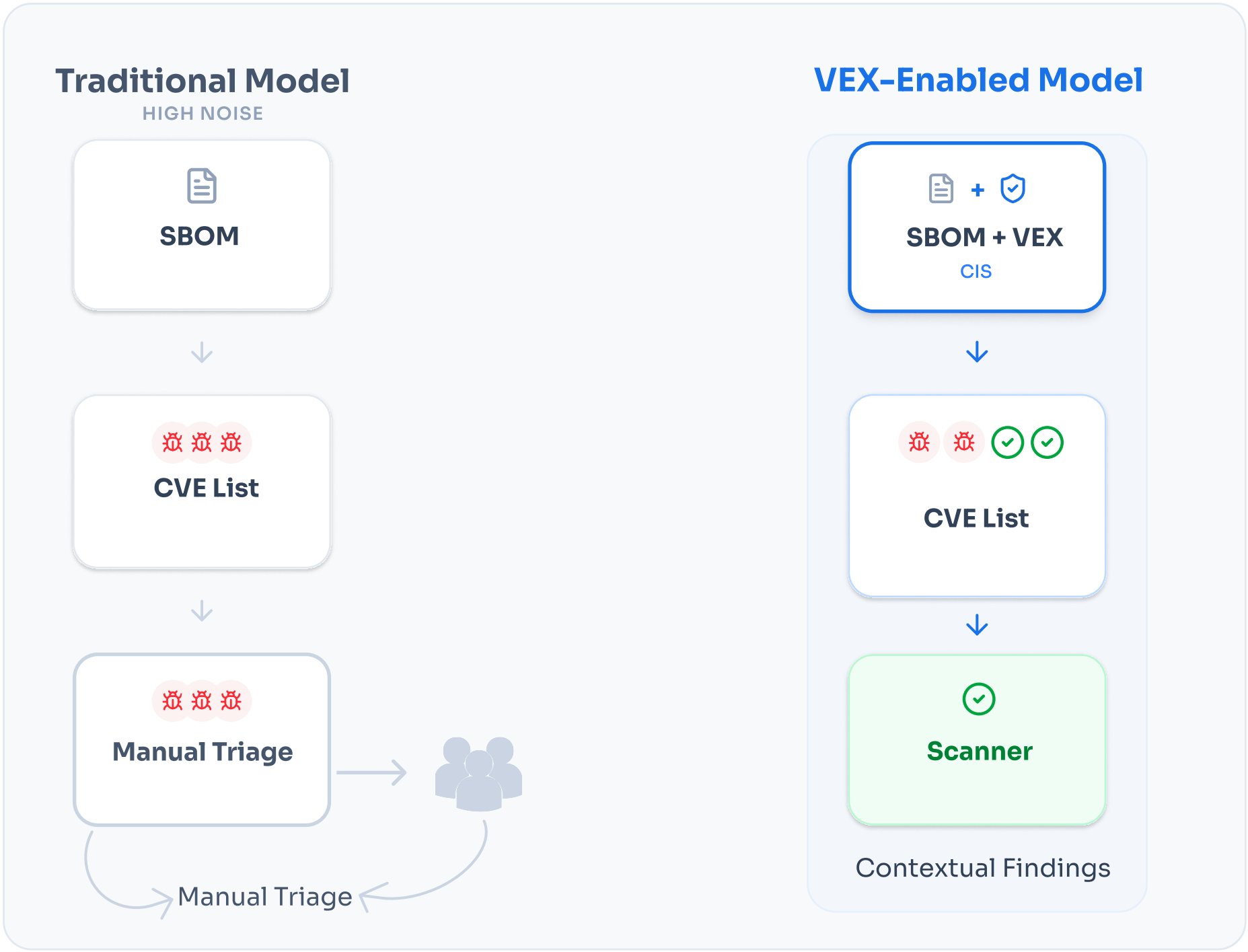
- **Not Affected**
- **Affected**
- **Fixed**
- **Under Investigation**

Each status is accompanied by a justification explaining why the vulnerability is or is not exploitable in the specific context of the product.

1. SBOMs answer what is included.
2. Scanners answer what vulnerabilities are known.
3. VEX answers which vulnerabilities matter, and why.

Together, they form a complete security signal.





CleanStart's Architectural Implementation of VEX

CleanStart implements VEX as part of the container build and publication process, not as a downstream workflow.

Context is preserved as an artifact, not recreated as a process.

Continuous maintenance of exploitability status

As vulnerabilities are fixed, reclassified, or newly discovered, CleanStart updates VEX information accordingly. Teams do not re-triage historical decisions. Exploitability remains current without manual intervention.

Compatibility with existing tooling

CleanStart VEX documents follow open standards and integrate with common scanners and platforms, including Trivy, Grype, and Docker Scout. Consumers ingest VEX data directly into existing workflows without changing tooling.

Operational Impact

Treating exploitability as an architectural property changes how teams operate.

- **Security teams** focus on real risk instead of alert volume
- **Developers** stop re-triaging the same vulnerabilities across builds
- **Platform teams** reduce policy friction and exception handling
- **Customers** receive clearer, defensible security signals
- **Auditors** see standardized, artifact-linked risk rationale

Vulnerability management becomes repeatable and defensible instead of reactive.

Relationship to SBOM Requirements

VEX does not reduce transparency. It makes transparency usable.

As SBOM adoption becomes mandatory across regulated industries, VEX provides the interpretive layer that allows vulnerability data to be consumed responsibly. Suppliers are able not only to disclose vulnerabilities, but to explain their relevance.

This is increasingly expected in mature supply chain security programs.

Key takeaway

Vulnerability scanners detect what exists.
Security decisions require knowing what is exploitable.

Without a standardized way to attach exploitability context to software artifacts, vulnerability management does not scale. VEX transforms vulnerability data from noise into signal by making context architectural.

References and Further Reading

- Vulnerability Exploitability eXchange (VEX)
- Minimum Elements for a VEX
- CycloneDX VEX Specification
- SPDX Vulnerability Exploitability Information
- NIST Secure Software Development Framework (SSDF)
- CISA and NIST Guidance on SBOMs and Vulnerability Disclosure