vibgrate

WHITEPAPER

Software risk & drift scoring: a methodology whitepaper

DriftScore, RiskScore, and DriftRisk™ — the factors, sources, and rationale

How Vibgrate measures maintainability drift (DriftScore) and security exposure (RiskScore) as separate axes, then derives DriftRisk™ as an executive headline. Discloses the evidence hierarchy (observed exploitation → predicted likelihood → capped severity → context), the source→role→weight mapping, the data-source robustness posture (OSV ∪ GHSA, EPSS, KEV, NIST LEV), and the exact published formulas — the credit-bureau model of open method, proprietary calibration.

Vibgrate July 10, 2026 15 min read

vibgrate.com/whitepapers/software-risk-and-drift-scoring-methodology

Source: github.com/vibgrate/cli

DOI: 10.5281/zenodo.21336304

Audience: security-literate engineering leaders and skeptical evaluators. **Posture:** disclosed factors, sources, and rationale; proprietary calibration constants withheld — the credit-bureau model. Everything you need to understand, audit, and challenge a Vibgrate score is here. The exact tuned weights and the breaking-change corpus stay proprietary.

Methodology tags covered: `driftscore-3.0`, `riskscore-1.0`, `driftrisk-1.1`. **Feed snapshot convention:** every score is a function of `(lockfile@commit, feeds@snapshot-date)` and stamped with both. **Scope:** this paper covers **dependency and runtime risk & drift derived from a project's lockfiles/manifests** (the packages, versions, runtimes, and known vulnerabilities they resolve to) — not application source-code SAST, secrets, or infrastructure misconfiguration. **Availability:** DriftScore is free (CLI); **RiskScore and DriftRisk are premium, Vibgrate Cloud-only** and require a completed scan ingestion (server-side security data + the blend).

1. Executive summary

Most vulnerability programs still triage by severity — sort the CVSS column, start at the top, work down. That method is now demonstrably the wrong default. The organizations that define these standards say so plainly: CVSS "is not a measure of risk" [2][3]. The public data supply it depends on is contracting: as of 15 April 2026 the U.S. National Vulnerability Database **prioritizes enrichment for** CVEs that appear in CISA's Known Exploited Vulnerabilities catalog, federal software, or software designated critical under Executive Order 14028 [5][6]. (All CVEs are still catalogued; lower-priority records simply are not scheduled for immediate enrichment.) NIST does not quantify the prioritized share; a Cloud Security Alliance research note estimates those categories at roughly **15–20% of projected annual CVE volume** — an industry estimate, not a NIST statistic [25]. A scoring model that assumes a fully enriched NVD is already built on sand.

Vibgrate takes a different position, aligned with the current evidence-first consensus (CISA KEV → EPSS → SSVC, now extended by NIST's LEV work) [9][11][15]. We publish **three** numbers, not one:

DriftScore (`driftscore-3.0`) — maintainability and currency. How far the stack has drifted from supported, current baselines.

RiskScore (`riskscore-1.0`) — security and business exposure. The probability and consequence of harm right now.

DriftRisk™ (`driftrisk-1.1`) — a derived executive headline, a pure function of the two axes above.

DriftScore and RiskScore measure different things and must stay separate: a stale-but-safe stack and a current-but-exploited stack are different problems and deserve different numbers. This paper discloses the factors behind each score, the data sources, the reliability of each source, and — most importantly — the *rationale* for how much weight each input earns. The single most consequential design choice is that **observed exploitation acts as an override, not a weight**: a live exploit cannot be averaged away by a hundred quiet findings.

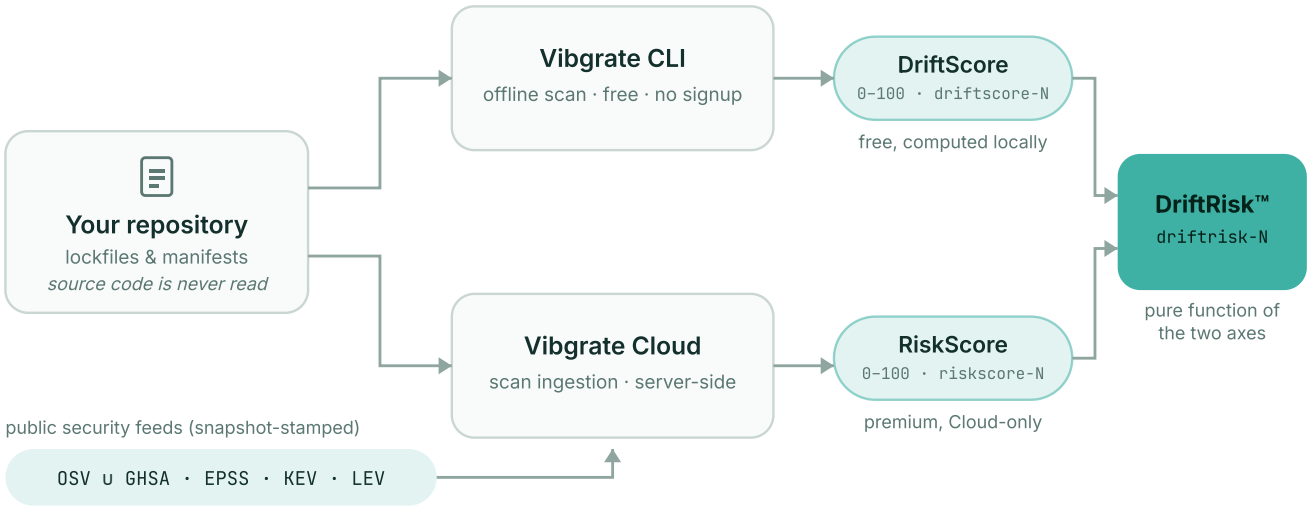

Figure 1 — How the scores are produced. The Vibgrate CLI computes DriftScore locally from lockfiles and manifests — source code is never read. Vibgrate Cloud computes RiskScore from an ingested scan plus snapshot-stamped public security feeds (OSV ∪ GHSA, EPSS, KEV, NIST LEV). DriftRisk™ is a pure function of the two published axes.

2. The two-axis thesis

Why two axes and not one

A single composite "security score" forces two unlike quantities into one number and loses both. Consider two services. Service A runs a fully current stack with one dependency carrying an actively exploited CVE. Service B is three major versions behind across the board, on an end-of-life runtime, with no known exploit. Collapse each to one number and they can land in the same band — yet the correct action differs completely. A needs an emergency patch today; B needs a planned modernization quarter. Vibgrate keeps them on separate axes so the number tells you *which* problem you have.

SCORE	AXIS	SCALE (0 = BEST)	TAG
DriftScore	Maintainability / currency	0 current → 100 max drift	`driftscore-N`
RiskScore	Security & business exposure	0 safe → 100 max risk	`riskscore-N`
DriftRisk™	Combined executive headline	0 → 100 (more = more pressure)	`driftrisk-N`

The two axes are deliberately inverted-proof against each other: both run 0 (good) to 100 (bad), but they answer different questions, so a reader who sees `Drift 74 · Risk 22` immediately understands "behind, but not under attack." DriftRisk is the third, derived number — a headline for executives who want one figure, always shown beside its two constituents so the detail is never hidden behind the blend.

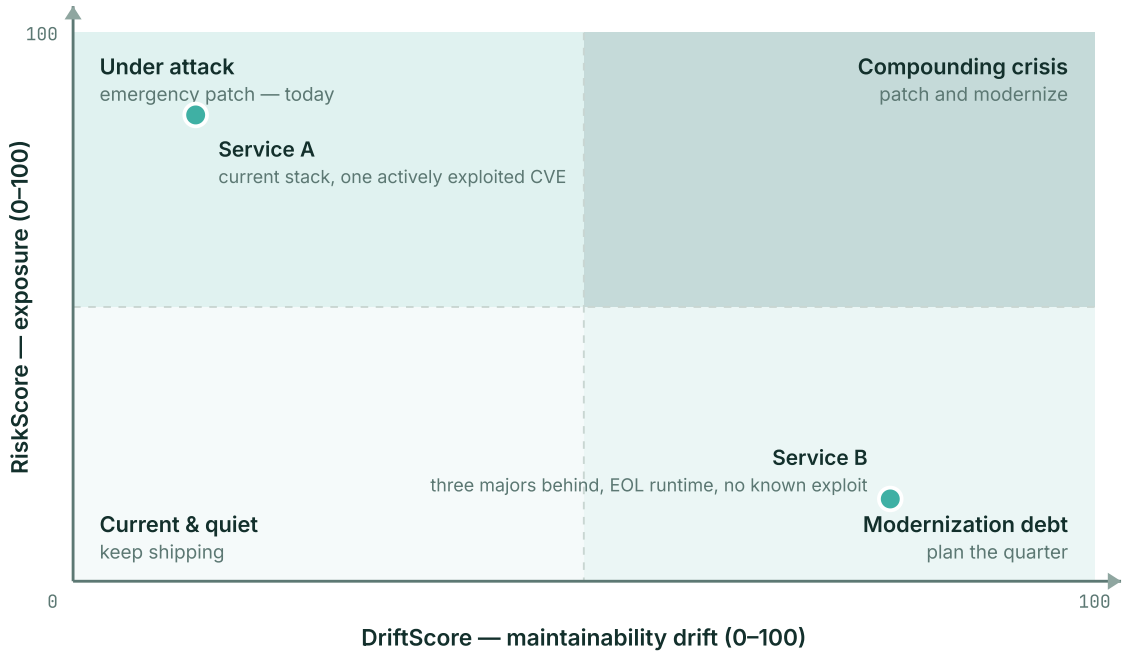

Figure 2 — The two-axis thesis. A single composite score can land Service A and Service B in the same band; on the plane they sit in opposite corners and demand different actions — an emergency patch versus a planned modernization quarter.

Trademark and openness

DriftRisk is a trademark of Vibgrate; the **algorithm is open source** and specified in full in section 7 and in the public scoring specification, `SCORING-METHODOLOGY-PUBLIC.md`. *DriftScore* and *RiskScore* are not trademarked. Every emitted score carries its methodology tag, so two numbers are only ever compared across matching methodologies — a dashboard never draws a trend line across a methodology change.

3. The evidence hierarchy for risk

The core claim of RiskScore is that vulnerability inputs are not equally trustworthy, and a defensible model must rank them by evidentiary strength:

Figure 3 — The evidence hierarchy for risk. Stronger evidence earns a stronger role: observed exploitation overrides, exploit likelihood weights, severity is capped, and context only scales what the evidence established.

Tier 1 — Observed exploitation (strongest, KEV only). CISA's KEV catalog is the authoritative record of vulnerabilities exploited in the wild; a KEV entry requires reliable evidence that malicious code execution occurred [11]. This is *confirmed fact*, and it is the **only** signal Vibgrate treats as a hard override. When exploitation is observed, prediction and severity become secondary: you already know.

A deliberate distinction — LEV is *not* observed exploitation. NIST's LEV metric (CSWP 41, May 2025) is a **proposed, probabilistic** metric that estimates, from a CVE's EPSS time series, the cumulative probability a vulnerability has *ever* been exploited — a conservative, lower-bound estimate designed partly to measure how comprehensive KEV is [9][10]. NIST itself describes it as proposed, with further validation needed. Vibgrate therefore treats LEV as **strong probabilistic evidence / a conservative historical-exploitation estimate** — it can escalate priority and flag likely gaps in KEV — but **only KEV floors the score.** Grouping LEV with KEV as "observed exploitation" would overstate it.

Tier 1b — CISA's own federal model now agrees (BOD 26-04). On 10 June 2026 CISA issued BOD 26-04, "Prioritizing Security Updates Based on Risk," superseding the flat KEV-deadline model of BOD 22-01 with a **four-variable risk model** — asset exposure, KEV status, exploit automatability, and technical impact [13][14]. KEV is no longer a standalone prioritisation rule; it is one high-confidence exploitation signal inside a broader risk decision that also weighs exposure, automation, and impact — precisely the structure of Vibgrate's evidence hierarchy plus its context tier (§Tier 4).

Tier 2 — Predicted likelihood. EPSS (FIRST) is a daily 0–1 probability estimate that a CVE will be exploited in the wild within the next 30 days [7][8] — a *forward-looking* probability, in contrast to LEV's *historical/cumulative* estimate. RiskScore consumes FIRST's **current EPSS feed**, which serves the EPSS v4 model family (published from 17 March 2025); we do not pin an older model version. It is the right tool for the far larger set of CVEs that are *not* in KEV. (FIRST's own current model page illustrates the efficiency: at an *example* threshold of 0.1, EPSS required ~2.7% remediation effort for ~63.2% coverage at ~65.2% efficiency, versus CVSS ≥ 7 requiring ~57.4% effort for ~82.2% coverage at ~3.96% efficiency — FIRST notes the threshold is illustrative, not a universal recommendation [7].) Where we cite EPSS performance we use FIRST's current documentation; we deliberately do **not** carry forward the older EPSS v3-era benchmark figures (e.g. the 0.779 AUPRC / "82% improvement" results) as if they were v4-specific, and treat any such figure as historical v3 evidence only.

Tier 3 — Severity, capped. CVSS base score describes the technical severity of a vulnerability if exploited — its worst-case mechanical impact. FIRST and NVD are explicit that CVSS measures severity, not full risk [2][3]. Severity is a useful *ceiling* on consequence, but on its own it over-selects: it cannot tell you whether anyone is actually attacking the thing. Vibgrate therefore uses CVSS to shape and cap the consequence term, never as the primary sort key.

Tier 4 — Context. SSVC (CMU SEI + CISA) and reachability analysis refine the ranking with situational facts: is the affected product mission-critical, is the attack automatable, is the vulnerable function actually invoked [15][18][16][17]. Context can raise or lower priority but never manufactures exploitation evidence that isn't there.

Why CVSS-severity-first triage is obsolete. Severity-first assumes the CVSS column is (a) complete, (b) accurate, and (c) a proxy for risk. All three assumptions have failed. NVD enrichment — the step that attaches CVSS vectors to CVEs — no longer keeps pace: a large share of 2024–2025 CVEs went unenriched, and from April 2026 NVD prioritizes enrichment for KEV/federal/EO-critical CVEs [5][6]. CVSS v4.0, introduced partly to correct v3.1's "everything is 9.8" score inflation, is not directly comparable to v3.x and commonly co-exists with it during transition, so a single CVE can carry divergent scores from different sources [4]. And FIRST's own guidance is that the number was never a

risk measure to begin with [1]. A model that leads with severity inherits every one of these defects. Vibgrate leads with exploitation evidence and lets severity cap the tail.

4. Source → role → weight mapping (RiskScore)

The table below is the heart of the disclosure. For each RiskScore input it names the data source, that source's reliability, and — the part most models omit — the *role* the input plays: an **override/floor** (can set the score on its own), a **weight** (contributes proportionally), or a **multiplier** (scales another term). Exact calibration constants are proprietary; the roles and public floors are not.

INPUT	DATA SOURCE	RELIABILITY	ROLE
Observed exploitation	CISA KEV	Authoritative — confirmed in-the-wild [11]	**Hard override — floors score at 80**
RATIONALE — Confirmed exploitation is the strongest evidence tier. It must dominate, not blend — otherwise many low findings drown one real emergency.			
Historical-exploitation estimate	NIST LEV (composite)	Proposed, probabilistic, conservative lower bound [9]	**Strong likelihood signal (not an override)**
RATIONALE — Estimates whether exploitation was *likely observed* over time; escalates priority and flags KEV gaps. Composite probability = `max(EPSS, KEV, LEV)` [9]. Not confirmed exploitation, so it does not floor the score.			
Predicted likelihood	EPSS v4 (FIRST)	Strong, probabilistic; CVE-only, telemetry-biased [7][8]	**Primary weight (likelihood term)**
RATIONALE — The best available *forward-looking* (30-day) signal for the vast non-KEV majority. Daily-updated, efficient relative to severity-only triage.			
Technical severity	CVSS base (via OSV/GHSA advisories)	Severity, *not* risk; enrichment shrinking [1][5]	**Capped weight (consequence ceiling)**
RATIONALE — Bounds worst-case impact. Capped so a high CVSS with negligible EPSS and no KEV can't dominate.			

INPUT	DATA SOURCE	RELIABILITY	ROLE
Exposure / lifecycle	endoflife.date, deprecation flags	Deterministic, well-sourced	**Floor**
RATIONALE — An EOL/unsupported component has no patch path; a floor prevents it reading healthy.			
Business / mission weight	Scope metadata (env, business unit, data sensitivity)	Customer-supplied context	**Multiplier (SSVC-style)**
RATIONALE — A vulnerability in a public-facing, sensitive service matters more than the same one in a sandbox. Scales, never invents, risk.			
Reachability *(where available)*	Call-graph analysis	Reduces false positives; language-limited [16][17]	**De-emphasis modifier**
RATIONALE — An unreachable vulnerable function is lower priority. Lowers noise; never used to *raise* a score above its evidence.			

Independent findings combine with a **probabilistic OR**, so many low-risk findings never sum their way past a single actively exploited one. Bands: **0–19 low · 20–49 moderate · 50–79 high · 80–100 critical**. Every RiskScore decomposes into its top contributing findings — if it can't be explained, it isn't published.

The override, restated because it is the crux. KEV membership — confirmed exploitation — does not add points; it *floors* the score at 80. LEV, being a *probabilistic estimate* rather than confirmed exploitation, strongly informs the likelihood term and can escalate priority, but it does **not** trigger the override — only observed exploitation (KEV) does. This mirrors the evidence hierarchy directly: no quantity of severity arithmetic, and no probability estimate, should be able to talk you out of an observed, in-the-wild exploit. Weights are for graded contributions; overrides are for facts.

5. Data-source robustness

A risk model is only as good as its feeds, so Vibgrate treats source selection as a first-class methodology decision.

Vulnerabilities: OSV ∪ GHSA, not NVD alone. No single database is complete. Safeguard's vendor-published comparison reports two related analyses. In a **1,000-vulnerability sample from 2024–2025**, coverage was approximately NVD ~89%, OSV ~93%, GHSA ~87%, and ~98% for the union of all three [19]. In a **separate comparison across the top-5,000 most-downloaded packages** (npm, PyPI, Maven Central) in the same article, OSV and GHSA carried affected-version ranges and fix versions far more often than NVD (e.g. OSV ~98% affected-range and ~94% fix-version availability, versus NVD ~71% and ~43%) [19]. These are two different samples, and — as vendor-published research from a single source — should be read as *indicative, not definitive* industry-wide measurements. The direction is corroborated by structural facts: OSV uses package-native (PURL) matching rather than NVD's CPE strings, and OSV.dev aggregates 30+ ecosystem sources under a common schema (GitHub Security Advisories, PyPA, RustSec, Global Security Database and more) [20][21]. For an open-source dependency scanner, package-native coverage, version-range precision, and freshness matter more than any single feed's brand. Vibgrate therefore sources from **OSV unioned with GHSA**, and treats NVD's CVSS as one severity input among several rather than the spine — a deliberate hedge against the enrichment contraction described in section 3 [5][6].

Exploitation & likelihood: EPSS + KEV + LEV composite. RiskScore consumes EPSS for predicted likelihood, KEV for observed exploitation, and follows NIST's Composite Probability construction — `max(EPSS, KEV-signal, LEV)` — so the strongest available exploitation signal wins rather than being diluted by weaker ones [9][10]. This is a direct implementation of the evidence hierarchy in feed form.

Reproducibility: every feed is snapshot-stamped. Because KEV, EPSS, and the advisory feeds all change daily, a score is meaningless without a timestamp. Vibgrate stamps each feed's snapshot date into the score envelope, so a RiskScore is reproducible as `(advisories@date, EPSS@date, KEV@date)`. Two scores are only ever compared across matching methodology tags *and* comparable snapshot windows.

6. DriftScore methodology in brief (`driftscore-3.0`)

DriftScore answers a different question: how far has this stack drifted from current, supported, maintainable baselines? It is version/time distance, not consequence — a DriftScore of 0 says nothing about whether a current package has a CVE (that is RiskScore's job). Fuller detail lives in the public scoring specification, `SCORING-METHODOLOGY-PUBLIC.md`; the essentials:

Libyear backbone. Each dependency is scored 0–100 as a blend of *time* distance and *version* distance. The time term rests on the **libyear** — the calendar time between the version in use and the latest stable release — the established, ecosystem-comparable freshness unit introduced by Cox et al. at ICSE 2015 [22][23]. Major-version counts alone are not comparable across ecosystems with different release cadences, so time is the backbone (weight 0.55) and version distance the complement (0.45); version-only scans fall back to the version term and are branded **Estimated** (`~NN`).

Tail-surfacing aggregation. A mean hides the one catastrophic dependency behind hundreds of fresh ones. Portfolio drift therefore blends a weighted mean with a p95 term and an unsupported-share term (`0.5·weightedMean + 0.3·p95 + 0.2·unsupported_share·100`), and unsupported/EOL components hit hard floors that cannot be averaged away. Four data-quality guards handle real-registry pathologies (canary "latest" versions, version-scheme jumps, squatted builtin stubs, high-cadence SDKs).

Verified vs Estimated provenance. Every score is `f(lockfile@commit, snapshot@date)`. **Verified** means release-date data was available — online *or* from a vendored dated snapshot, so offline is *not* Estimated. **Estimated** means version-only, shown with a leading `~`. A separate `confidence` field reports the fraction of dependencies that resolved, so a low-coverage scan cannot masquerade as clean. DriftScore bands: **0–30 low · 31–60 moderate · 61–100 high**.

7. DriftRisk™ blend (`driftrisk-1.1`)

DriftRisk is a single number for executives — "how much pressure is this codebase putting on the team to act?" — computed *purely* from the two published axes, never feeding back into either.

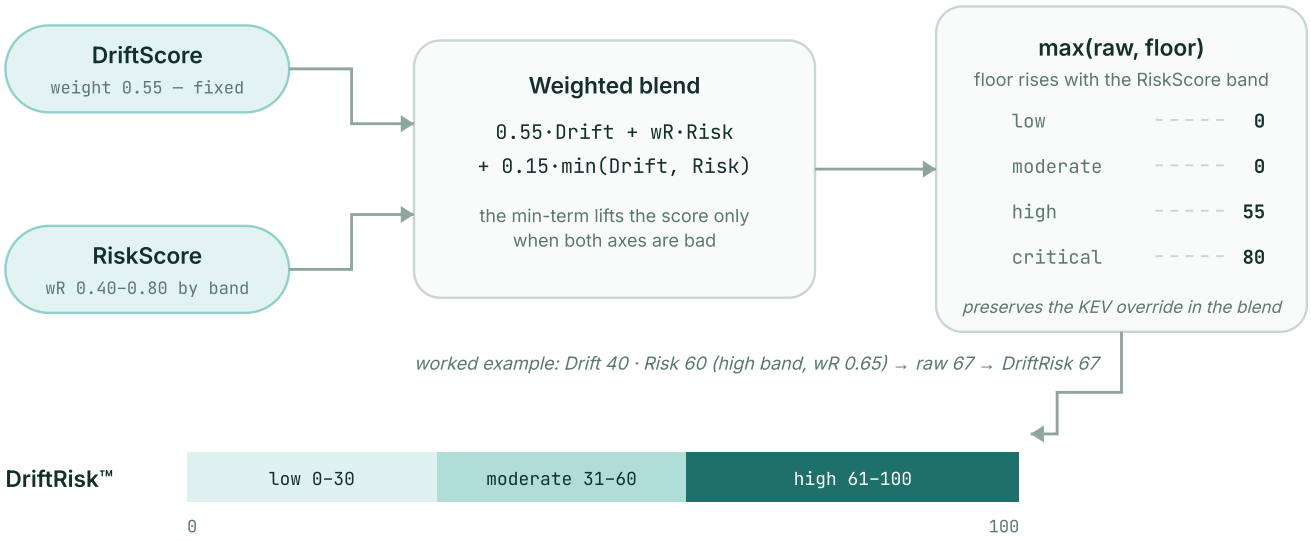

Figure 4 — The DriftRisk™ blend (driftrisk-1.1). A fixed drift weight, a risk weight that rises with the RiskScore band, a danger-zone amplifier that fires only when both axes are bad, and a floor ladder that preserves the KEV override at the blended level. Worked example as computed in section 7.

```
band      = riskBand(RiskScore)                    # low | moderate |
high | critical
wR        = { low: 0.40, moderate: 0.50, high: 0.65, critical: 0.80 }
[band]
floor     = { low: 0,    moderate: 0,    high: 55,   critical: 80 }
[band]
raw       = min(100, 0.55·DriftScore + wR·RiskScore +
0.15·min(DriftScore, RiskScore))
DriftRisk = min(100, max(raw, floor))
```

RISKSCORE BAND	RISK WEIGHT `WR`	FLOOR
low (0–19)	0.40	0
moderate (20–49)	0.50	0
high (50–79)	0.65	55
critical (80–100)	0.80	80

Three design properties matter for a "respected-methodology" number:

Evidence-tiered dynamic weighting. Risk's weight grows with the RiskScore *band* — from 0.40 at low risk to 0.80 at critical — so a serious security posture emphasises risk instead of being averaged down. Because the band is itself evidence-weighted (section 3: KEV can force the critical band, high LEV can contribute strong probabilistic pressure without acting as an override, EPSS drives likelihood, and CVSS is only a capped consequence input), tiering on the band means DriftRisk emphasises *exploitation evidence and exploit likelihood, not raw severity*. The `0.15·min(Drift, Risk)` term is an additional danger-zone amplifier — it only lifts the score when *both* axes are bad.

A floor ladder that generalises the KEV override. The `floor[band]` term is the critical correction, and the critical-band floor (80) preserves the practical effect of the KEV override at the blended DriftRisk level. Without it, an actively exploited CVE (RiskScore ≥ 80) on an otherwise-current stack (low Drift) would blend to a calm "moderate" — a live security emergency reading green because upgrade debt is low. The floor lifts it to **80 — the top urgency band for DriftRisk** (DriftRisk has only low/moderate/high bands; the RiskScore "critical" band is what *triggers* the floor, not a DriftRisk band label). This mirrors the KEV hard-floor inside RiskScore: observed exploitation overrides the blend, it does not average into it.

Monotonic and sortable. Drift weight is fixed at 0.55 and never reallocated; the risk weight and floor only ever *rise* with the band. So DriftRisk is provably non-decreasing in both inputs — nothing gets *safer* by getting worse on an axis — and is a single global function `f(Drift, Risk)`, making it a legitimate sort key for "what needs attention first" while remaining comparable across a portfolio.

Pair for reading, scalar for ranking. Use the DriftRisk scalar for ranking and badges; show the `Drift · Risk` pair beside it for reading — `Drift 40 · Risk 60 · DriftRisk 67` — because the scalar orders a list and the pair explains any one row. The full breakdown is always one click away; a

space-constrained surface (a badge, a table cell) may show the scalar alone. CI gates should fire on the *axis* that matters (a Risk gate for security, a Drift gate for modernization budgets), not on the blended headline. DriftRisk bands: **0–30 low · 31–60 moderate · 61–100 high**.

Worked examples. `Drift 40 · Risk 60` (high band, wR 0.65) → `raw = 0.55·40 + 0.65·60 + 0.15·40 = 67` → **DriftRisk 67** — risk clearly leads (a flat 0.55/0.45 blend would have read 49). Now land one CVE in CISA KEV → RiskScore floors to the critical band (≥ 80): even at `Drift 12`, `raw = 6.6 + 64 + 1.8 = 72`, but the critical floor lifts it to `max(72, 80) = 80` — DriftRisk reads high, "patch now."

8. Limitations & honesty

No score is a guarantee, and a respected methodology states its own boundaries.

What the scores are not. RiskScore is not a breach predictor and DriftScore is not a bug count. A RiskScore of 0 means "no known, sourced exposure at this snapshot," not "safe." A DriftScore of 0 means "current," not "vulnerability- free." Neither number is a compliance attestation.

Probabilistic inputs stay probabilistic. EPSS is a 30-day probability trained on IDS/telemetry data; it scores CVEs only, and its signal is biased toward what sensors observe, so novel or quietly exploited issues can score low [7][8]. LEV is explicitly a lower-bound estimate [9]. We surface these as probabilities, never as certainties.

KEV is authoritative but not comprehensive. CISA itself does not recommend KEV as the sole triage criterion; it is a subset of all exploited vulnerabilities [11][12]. That is precisely why RiskScore also carries EPSS and the LEV composite rather than treating "not in KEV" as "not exploited."

Severity data is thinning. With NVD enrichment prioritized (not universal) from April 2026, CVSS coverage for the long tail of CVEs will decline; RiskScore's reliance on OSV/GHSA advisories is a hedge, but severity remains a partial input [5][6].

Reachability is not universal. Where call-graph reachability is available it lowers noise, but it is language- and ecosystem-limited and errs toward work-reduction, so absence of a reachability signal is not evidence of safety [16][17].

Coverage is reported, not assumed. The `confidence` field reports how much of the dependency set resolved. A low-confidence scan is labelled as such; we do not let partial data present as a clean bill of health.

Empirical validation is in progress. This paper documents the *design* rationale; a companion validation appendix (planned) will report back-testing of RiskScore's ranking against historical outcomes — e.g. how highly the model ranked CVEs *before* they entered CISA KEV, and calibration of the likelihood term against realised exploitation outcomes. Until then, the weightings are honestly labelled expert-and-evidence-informed, not empirically fitted.

9. Versioning & disclosure posture

Independent, explicit versioning. DriftScore, RiskScore, and DriftRisk carry separate methodology tags (`driftscore-3.0`, `riskscore-1.0`, `driftrisk-1.1`) that bump **only** when a formula or weighting changes — never for routine CLI releases. Dashboards refuse to trend across a tag change, so a methodology update can never masquerade as a movement in the underlying stack.

Snapshot pinning. Beyond the methodology tag, every score records the feed snapshot dates it was computed against, making any published number reproducible.

Open core, proprietary calibration — two distinct precedents. We publish the axes and their philosophy, every factor and data source, the formulas, the bands, worked examples, and the coverage/confidence semantics — the core scoring spec is openly available — while withholding only the exact calibration constants, the tuned weight vectors, and the breaking-change intelligence corpus. That posture draws on **two different precedents, which should not be conflated**:

OpenSSF Scorecard shows the value of a fully *transparent, check-based* score — its checks, its 0–10 range, and its risk-weighted aggregation weights are all public [24]. It is a precedent for **openness of method**, not for proprietary tuning.

Credit-bureau-style models show how *tuned calibration* can remain proprietary while the *factors and explanations* stay disclosed.

Vibgrate combines them: method and factors open (Scorecard-style), exact tuning proprietary (credit-bureau-style).

10. References

All URLs verified during research (July 2026). Primary/authoritative sources are cited for every non-obvious empirical claim; where a claim rests on a single secondary study it is flagged as such in-text.

1. FIRST — CVSS v4.0 Specification Document. https://www.first.org/cvss/v4.0/specification-document
2. NIST NVD — Vulnerability Metrics (CVSS; "CVSS is not a measure of risk"). https://nvd.nist.gov/vuln-metrics/cvss
3. FIRST — CVSS v3.1 User Guide (CVSS measures technical severity, not risk). https://www.first.org/cvss/v3.1/user-guide
4. "Fragmentation of CVSS scores in the NVD: a quantitative analysis of inconsistency across vulnerability scoring standards," *Computers & Security* (2026). https://www.sciencedirect.com/science/article/abs/pii/S0167404826001549
5. NIST — "NIST Updates NVD Operations to Address Record CVE Growth" (April 2026). https://www.nist.gov/news-events/news/2026/04/nist-updates-nvd-operations-address-record-cve-growth
6. Help Net Security — "NIST admits defeat on NVD backlog, will enrich only highest-risk CVEs going forward" (16 April 2026). https://www.helpnetsecurity.com/2026/04/16/nist-national-vulnerability-database-nvd-enrichment/
7. FIRST — EPSS Model. https://www.first.org/epss/model
8. FIRST — EPSS Data and Statistics. https://www.first.org/epss/data_stats
9. NIST — CSWP 41, "Likely Exploited Vulnerabilities: A Proposed Metric for Vulnerability Exploitation Probability" (May 2025). https://csrc.nist.gov/pubs/cswp/41/likely-exploited-vulnerabilities-a-proposed-metric/final — PDF: https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.41.pdf
10. Greenbone — "LEV: Demystifying the New Vulnerability Metrics in NIST CSWP 41." https://www.greenbone.net/en/blog/lev-demystifying-the-new-vulnerability-metrics-in-nist-cswp-41/
11. CISA — Known Exploited Vulnerabilities Catalog. https://www.cisa.gov/known-exploited-vulnerabilities-catalog
12. CISA — "Reducing the Significant Risk of Known Exploited Vulnerabilities" (BOD 22-01; now revoked/superseded by BOD 26-04). https://www.cisa.gov/known-exploited-vulnerabilities

13. CISA — BOD 26-04, "Prioritizing Security Updates Based on Risk" (10 June 2026; supersedes BOD 22-01 and 19-02; four-variable model: exposure, KEV, automatability, technical impact). https://www.cisa.gov/news-events/directives/bod-26-04-prioritizing-security-updates-based-risk

14. Tenable — "What is CISA BOD 26-04: Impact on vulnerability remediation" (summary of the four-variable risk model). https://www.tenable.com/blog/cisa-bod-26-04-FAQ-vulnerability-remediation-impact

15. CISA — Stakeholder-Specific Vulnerability Categorization (SSVC). https://www.cisa.gov/stakeholder-specific-vulnerability-categorization-ssvc

16. Snyk — Reachability analysis (product documentation). https://docs.snyk.io/scan-fix-and-prevent/fix/prioritize-issues-for-fixing/reachability-analysis

17. Dark Reading — "Reachability Analysis Pares Down Vulnerability Reports." https://www.darkreading.com/application-security/reachability-analysis-static-security-testing-overload

18. CMU SEI — "Prioritizing Vulnerability Response: A Stakeholder-Specific Vulnerability Categorization (Version 2.0)." https://www.sei.cmu.edu/library/prioritizing-vulnerability-response-a-stakeholder-specific-vulnerability-categorization-version-20/

19. Safeguard — "Vulnerability Database Comparison: NVD vs OSV vs GHSA" (study of 1,000 open-source vulnerabilities, 2024–2025). https://safeguard.sh/resources/blog/open-source-vulnerability-database-comparison

20. OSV — Data sources. https://google.github.io/osv.dev/data/

21. OSV — Open Source Vulnerabilities database. https://osv.dev/

22. J. Cox, E. Bouwers, M. van Eekelen, J. Visser — "Measuring Dependency Freshness in Software Systems," ICSE 2015. https://www.researchgate.net/publication/308833452_Measuring_Dependency_Freshness_in_Software_Systems

23. libyear — "A simple measure of software dependency freshness." https://libyear.com/

24. OpenSSF Scorecard — open, checks-based scoring methodology (checks, 0–10 range, and risk-weighted aggregation weights all public). https://scorecard.dev/ — source: https://github.com/ossf/scorecard

25. Cloud Security Alliance — research note on the NIST NVD enrichment triage change (industry estimate of the enriched CVE slice). https://labs.cloudsecurityalliance.org/research/csa-research-note-nist-nvd-enrichment-overhaul-20260429-csa/

> ***Legal, trademark & disclosure notes.*** *Trademark attributions, the open-method / proprietary-calibration posture, the "as is" no-warranty disclaimer, the third-party-data notice, and governing law for this and every Vibgrate whitepaper are set out in* ***Appendix A — Legal, Trademark & Disclosure Notes*** *— the single shared appendix reproduced with each paper (* `WHITEPAPER-LEGAL-NOTES.md` *).*

Appendix A — Legal, Trademark & Disclosure Notes

These notes apply to this whitepaper and to every Vibgrate whitepaper. They are maintained as a single shared appendix and reproduced with each paper.

Trademarks

DriftRisk™ is a trademark of Vibgrate. **Vibgrate®** is a registered trademark of Vibgrate. *DriftScore* and *RiskScore* are not trademarked. All other trademarks, service marks, and trade names appearing in this document (including, without limitation, CISA KEV, EPSS, CVSS, NIST, SSVC, OpenSSF Scorecard, OSV, GHSA, and libyear) are the property of their respective owners and are used solely for nominative and descriptive purposes to identify the data sources and methodologies referenced herein.

Open method, proprietary calibration

Vibgrate publishes the methodology, factors, data sources, formulas, bands, and rationale for its scoring systems. The exact tuned weights, calibration constants, and the breaking-change intelligence corpus remain proprietary. This posture follows established industry precedents for transparent yet commercially sustainable security scoring systems — the credit-bureau model of open method, proprietary calibration.

Public specification

The algorithm and formulas for DriftRisk™ (`driftrisk-1.1`) are specified in full in the accompanying whitepaper and in `SCORING-METHODOLOGY-PUBLIC.md`, which is made available under an open-source license (see the repository for current license terms). DriftScore and RiskScore methodology details are disclosed to the extent necessary to understand, audit, and challenge published scores.

No warranty or guarantee

The information in this whitepaper, and any scores produced by Vibgrate software, are provided **"as is"** and **"as available"** without warranty of any kind, express or implied, including without limitation the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. Vibgrate makes no representation or warranty that any score is accurate, complete, reliable, or fit for any

particular purpose. Scores are tools to assist human decision-making; they are not a substitute for professional security assessment, penetration testing, or legal review.

Third-party data sources

Vibgrate incorporates or references data from third-party sources (including, without limitation, CISA, FIRST, NIST, OSV, and GHSA). Vibgrate does not control, verify, or endorse the accuracy or completeness of such third-party data, and users are encouraged to consult the original sources directly. Each score is stamped with the feed snapshot dates it was computed from, so any figure can be traced back to the source data as it stood at computation time.

Governing law

This document, and any dispute arising out of or in connection with it, shall be governed by and construed in accordance with the laws of **England and Wales**, without regard to its conflict-of-law principles, and shall be subject to the exclusive jurisdiction of the courts of **England and Wales**.

vibgrate

OUR MISSION

Keep software — and the AI now writing it — current, understood, and free of known, avoidable risk.

Vibgrate measures how far a codebase has drifted from current, supported versions and gives teams the path to clear the known, avoidable risk. The Vibgrate CLI computes a 0–100 DriftScore from your lockfiles — offline, with no signup, and without reading your source. Vibgrate Cloud adds RiskScore, the DriftRisk™ headline, and governed remediation across a portfolio, and Vibgrate AI Context serves version-correct library docs, a committable code map, and drift signals to the AI tools writing your code.

Website	`vibgrate.com`
Vibgrate Cloud	`dash.vibgrate.com`
Open-source CLI & scoring spec	`github.com/vibgrate/cli`
Contact	`vibgrate.com/contact`

Try it with no install — `npx @vibgrate/cli scan` — or see every install method at vibgrate.com/cli.

DriftRisk™ is a trademark of Vibgrate. Vibgrate® is a registered trademark of Vibgrate. All other marks are the property of their respective owners.
© 2026 Vibgrate · vibgrate.com