

Beyond One-Hop Aliases: Measuring Identifier-Resolution Fragmentation in the OSV Vulnerability Graph Across Six Package Ecosystems

Cybersecurity Advances • Volume 3 • Issue 1 • 2026 • pp. 14–24

ATS PRESS Research Group

AlphaTech Scholarly Press, United States

Correspondence: Md Sazzad Hossain • md@hossain.tech • ORCID 0009-0009-4948-1179

Received 2026-10-02 • Accepted 2026-10-02 • Published 2026-10-02

© 2026 ATS PRESS Research Group. Published by AlphaTech Scholarly Press under CC BY 4.0.

Abstract

Background: Open-source vulnerability records connect source-native identifiers through aliases. The Open Source Vulnerability (OSV) specification treats aliases as symmetric and transitive, but consumers that stop at one record's directly listed identifiers may recover an incomplete equivalence set.

Objective: This study measures the difference between direct alias lists and transitive closure in a frozen, cross-ecosystem OSV snapshot.

Methods: We acquired the official OSV archives for PyPI, npm, Maven, Go, crates.io, and RubyGems on 1 October 2026. A deterministic, test-driven pipeline deduplicated records, built an undirected alias graph, and measured closure completeness, explicit reciprocity, package and ecosystem breadth, and publication span. A sensitivity analysis added related links as weak associations without treating them as equality claims.

Results: The corpus contained 280,270 unique records, 345,225 identifier nodes, 94,492 alias edges, and 267,681 components. Transitive alias debt affected 334 records (0.119%), or 0.463% of records in non-singleton components. Explicit record-to-record declarations were reciprocal in 28,740 of 28,858 cases (99.591%). Debt occurred in 101 components. One 361-node component accounted for 180 debt-bearing records; excluding it reduced prevalence to 0.055%. Adding related edges merged 276 components and increased the largest component from 361 to 1,556 nodes.

Conclusions: OSV aliases were highly reciprocal and usually sufficient for one-hop resolution, but a small, concentrated set of components required closure. Consumers should implement typed graph traversal, preserve provenance, and keep alias equality separate from weaker relatedness.

Keywords: OSV; vulnerability identifiers; alias resolution; vulnerability databases; software supply chain; graph analysis

1. Introduction

Vulnerability-management systems rarely receive one identifier for one self-contained record. A software flaw may be represented by a CVE identifier, a GitHub Security Advisory identifier, and one or more ecosystem-specific advisory identifiers. Matching these records matters because affected-version ranges, remediation links, severity data, and publication times can be distributed across sources. If a tool treats different identifiers as unrelated, it may create duplicate tickets or leave evidence attached to only one record. If it merges identifiers too aggressively, it may combine distinct vulnerabilities.

The Open Source Vulnerability format addresses this problem with typed relationships. Its aliases field identifies records that describe the same vulnerability. The specification states that aliases should be considered symmetric and transitive [1]. The related field has weaker semantics: related identifiers are symmetric but not transitive [1]. These rules imply a graph model. A consumer that receives record A, follows only the identifiers directly listed by A, and stops may miss C when A aliases B and B aliases C. The omission does not necessarily indicate a defective source record because transitivity is a property consumers are expected to honor. It does, however, expose a practical difference between direct-list lookup and equivalence-class resolution.

Most empirical work on vulnerability-data quality has examined descriptions, affected products, dates, severity values, labels, or disagreement across repositories [3–6]. Recent work has integrated OSV, GitHub Advisories, and NVD data in a knowledge graph [7]. These studies establish that vulnerability metadata can be incomplete or inconsistent and that graph integration can help practitioners. They do not quantify the one-hop alias-resolution gap within OSV itself at cross-ecosystem scale.

This study audits that gap in a frozen snapshot of six official OSV ecosystem archives. The research questions concern graph size and identifier-family structure, one-hop closure completeness, explicit reciprocity, package and ecosystem breadth, publication span, and sensitivity to the inclusion of related links. The analysis is deliberately descriptive. It measures recorded relations and does not infer exploitability, production exposure, operational loss, or whether every observed alias is substantively correct.

The study makes four contributions. First, it provides a protocol-frozen measurement of 280,270 unique OSV records and 345,225 identifier nodes. Second, it distinguishes direct alias declarations from the transitive closure required for resolution. Third, it quantifies how the result is concentrated in a small number of components rather than presenting only a corpus-wide average. Fourth, it publishes a deterministic pipeline, tests, source hashes, derived tables, and figure code suitable for independent reproduction.

2. Background and related work

2.1 OSV relationship semantics

OSV is a machine-readable vulnerability format and aggregation service designed around package ecosystems and affected-version ranges. Official per-ecosystem archives allow consumers to obtain complete snapshots without issuing one API query per package [2]. Every record has a source-native id. The optional aliases list can connect that identifier to identifiers in other databases.

The distinction between aliases, upstream, and related is important. Under the frozen OSV specification, aliases describe the same vulnerability and should be considered symmetric and transitive [1]. An upstream link is transitive but directional. A related link covers associations that do not meet the strict alias or upstream definition and is symmetric but not transitive [1]. Collapsing all three relations into one equivalence relation would change their intended semantics.

This paper therefore treats alias edges as equality claims for graph closure and keeps related edges out of the primary graph. The related-edge analysis asks only how graph topology changes if those weaker links are traversed. It does not relabel them as aliases.

2.2 Vulnerability-data quality

Dong et al. compared public vulnerability reports with standardized NVD entries and found substantial inconsistencies in affected-version information [3]. Anwar et al. conducted a broad quality assessment of NVD fields, including publication dates, affected applications, severity scores, and classifications [4]. Jiang et

al. compared open-source vulnerability repositories and documented inconsistencies across them [5]. Croft et al. showed that data-quality problems in vulnerability datasets can propagate into learning-based security research [6].

These findings motivate careful treatment of any aggregated vulnerability corpus, but the current study evaluates a narrower mechanism. It does not decide which source has the correct description or score. It asks whether an identifier resolver that reads only one record's explicit aliases obtains the full equivalence class implied by all alias declarations in the frozen snapshot.

Boles et al. integrated NVD, GitHub Advisories, and OSV into a vulnerability knowledge graph [7]. Their study demonstrates the value of joining heterogeneous sources for practitioner queries. The present work differs in source scope and estimand. It analyzes the internal topology of OSV alias declarations, includes alias-only nodes even when no corresponding record appears in the six archives, and measures the exact loss from stopping after one hop.

Practitioner research also shows why seemingly small metadata gaps matter. NVD users report missing or incorrect entries, incomplete descriptions, and the need to cross-check sources [8]. Alias resolution is one part of that burden. A resolver that preserves relation type and computes reviewable closure can reduce manual identifier chasing without assuming that every connected record should be merged automatically.

3. Methods

3.1 Protocol and snapshot

The protocol was frozen at 2026-10-01T20:47:17Z and hashed before data acquisition. Its SHA-256 is 001f88e0a454ca37daf8be8285c6eab4bd8291a48b472e007460cca963729399. After the first run exposed an unusually large component, a dated amendment added concentration diagnostics without changing the corpus, graph definition, or primary estimands. The amendment SHA-256 is b65c09b3adb8073d29ebac2c744630d40aa771c5bba13adf9d9e829e3c0989a0.

We downloaded the official all.zip archives for PyPI, npm, Maven, Go, crates.io, and RubyGems on 1 October 2026 [2]. The pipeline stored the final URL, HTTP validators, acquisition timestamp, size, and SHA-256 for each archive. It verified ZIP CRCs and parsed every JSON entry. Table 1 reports the archive-level inventory.

Table 1. Frozen OSV archive inventory.

Archive	Parsed records	Compressed size (MB)
PyPI	25,995	35.33
npm	229,753	217.30
Maven	7,160	10.34
Go	9,421	11.95
crates.io	2,870	3.52
RubyGems	5,395	5.04

Archive membership can overlap. Records were grouped by id across the six archives. Identical canonical JSON payloads were deduplicated. The protocol specified deterministic resolution for divergent payloads by latest parseable modified timestamp and canonical hash tie-break, but no divergent record IDs occurred in this snapshot. The final union contained 280,270 records.

3.2 Graph construction

Every unique record ID and alias string became an identifier node. For each record A and each value B in its aliases list, the primary graph added an undirected edge A–B. Blank identifiers, self-links, and duplicate edges were rejected or collapsed as applicable. Isolated record IDs remained singleton components. Alias-only identifiers were retained even if no record with that ID appeared in the six selected archives.

For reciprocity, the pipeline also preserved directed declarations. $A \rightarrow B$ meant that record A explicitly listed B. Reciprocity was evaluated only when B was also a record-backed identifier in the frozen corpus. This avoids treating the absence of an unavailable record as an asymmetric declaration.

Identifiers were grouped descriptively by prefix: CVE, GHSA, GO, PYSEC, RUSTSEC, NPM, RUBYSEC, DSA/DLA, and other. This classification does not assign ownership where prefixes are ambiguous.

3.3 Closure metrics

For each record-backed identifier, the direct one-hop set was the record ID plus its explicitly listed aliases. The closure set was the complete connected component in the primary alias graph. One-hop completeness was

$$|\text{direct one-hop set} \cap \text{closure set}| / |\text{closure set}|.$$

A value below 1 indicated transitive alias debt: a one-hop resolver would miss at least one identifier reachable through another alias declaration. A secondary completeness measure limited the denominator to record-backed identifiers. Breadth beyond one hop was also summarized by the maximum shortest-path distance from each record to other nodes in its component.

Component-level measures included total nodes, record-backed IDs, identifier families, distinct normalized (ecosystem, package name) pairs, distinct ecosystems, and publication span. Package names underwent Unicode NFKC normalization and whitespace trimming while retaining case. Publication span was the interval between the earliest and latest parseable record published timestamps in components with at least two record-backed IDs. It is not a measure of defect discovery or coordinated-disclosure delay.

3.4 Sensitivity analyses

Four sensitivity analyses were prespecified. The first excluded divergent IDs, which did not change the corpus because none were observed. The second used record-backed closure completeness. The third added related edges and measured component merging without treating relatedness as equality. The fourth case-folded package names to test package-breadth sensitivity. Singleton components remained in population totals but were excluded from non-singleton completeness summaries.

The first run showed one 361-node component, prompting the protocol amendment. The definitive run therefore also reported debt-component count, the share of debt in the largest component, prevalence after excluding that component, and maximum breadth values. These diagnostics are post hoc and do not replace the primary results.

3.5 Statistical and reproducibility plan

The archives form a finite observed snapshot rather than a probability sample. We therefore report exact counts, proportions, and linear quantiles without confidence intervals or null-hypothesis tests. Every denominator is stated. No threshold, model, or stratum was selected from outcome performance.

Implementation used Python 3.13 and test-first development. Seventeen tests cover parsing, canonicalization, Unicode normalization, duplicate handling, deterministic variant resolution, graph closure, shortest-path distance, reciprocity, package breadth, publication spans, relation-type separation, and semantic hashing. Sorted iteration and deterministic tie-breaking were used throughout. A cached

rerun reproduced the same semantic-result SHA-256:
 efc9a8c123ac03d322daf6f93d72f5b114255f8e2ea7e9d4a361e5af3c004186.

4. Results

4.1 Corpus and graph structure

The 280,270 records yielded 345,225 identifier nodes and 94,492 unique undirected alias edges. The primary graph had 267,681 connected components. Most components were singletons. Among 72,110 record IDs in non-singleton components, the median and 95th-percentile one-hop completeness were both 1.0. Figure 1 shows the strongly right-skewed component-size distribution.

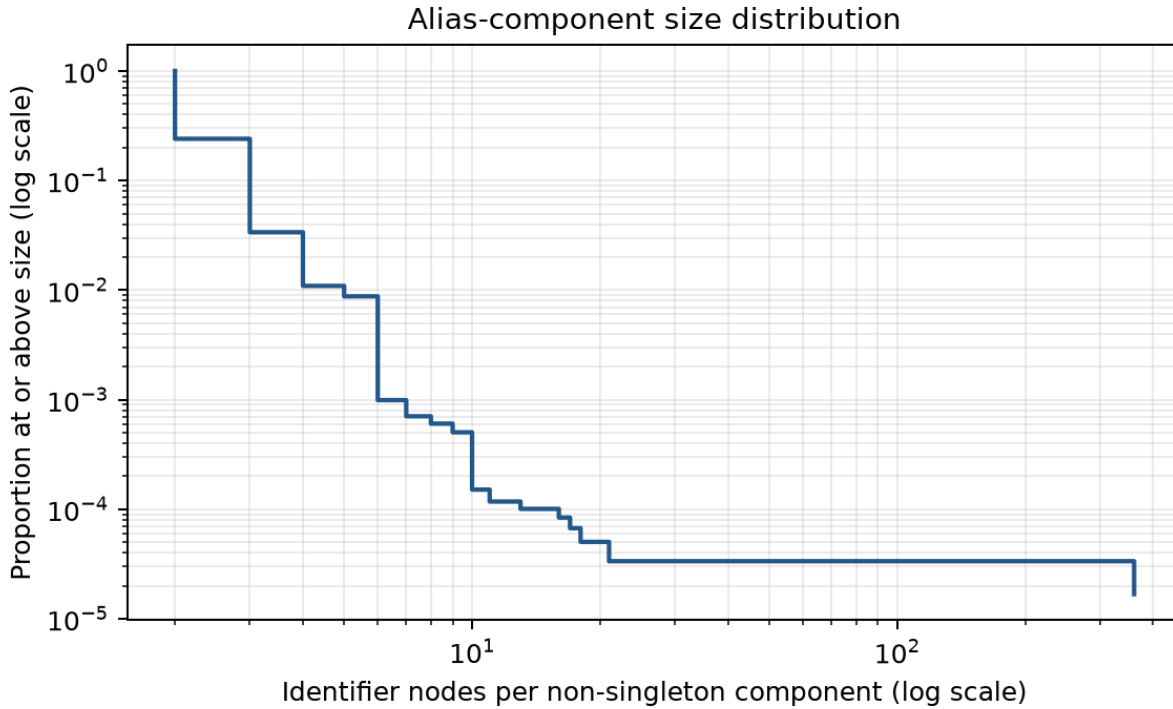


Figure 1. Complementary cumulative distribution of non-singleton alias-component sizes.

The largest component contained 361 identifier nodes and 180 record-backed IDs. The next-largest component had 21 nodes. Across all components, median and 95th-percentile package breadth were both one, but the maximum was 406 packages. A component could span as many as ten affected ecosystems. Only 352 of 267,681 components (0.1315%) included affected packages from more than one ecosystem.

4.2 One-hop completeness

A total of 334 records had one-hop completeness below 1. This equals 0.1192% of all records and 0.4632% of record IDs in non-singleton components. Debt appeared in 101 components. Among 10,986 components containing at least two record-backed IDs, 10,885 (99.081%) had every record explicitly closure-complete.

The debt distribution was highly concentrated. The 361-node largest component contained 180 debt-bearing records, or 53.89% of all debt-bearing records. Each of those records directly exposed three of 361 component identifiers, producing completeness 0.00831. Excluding the largest component left 154 debt-bearing records among 280,090 records, a prevalence of 0.0550%. Figure 3 displays the full completeness distribution among debt-bearing records.

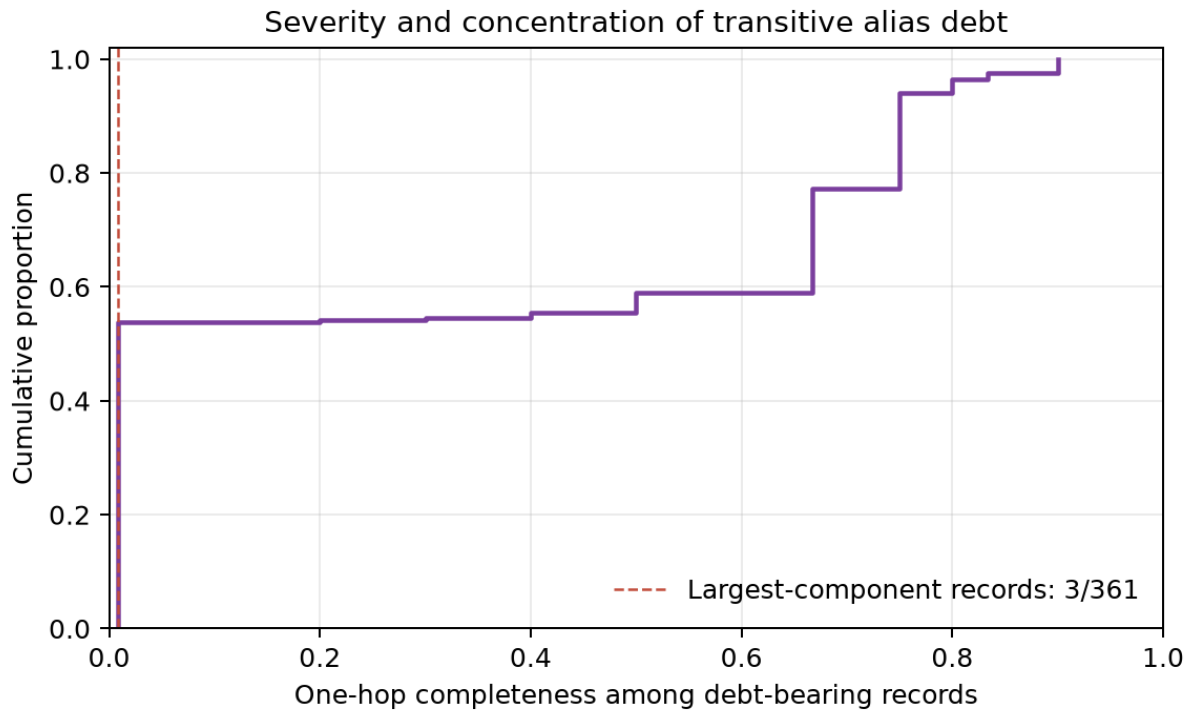
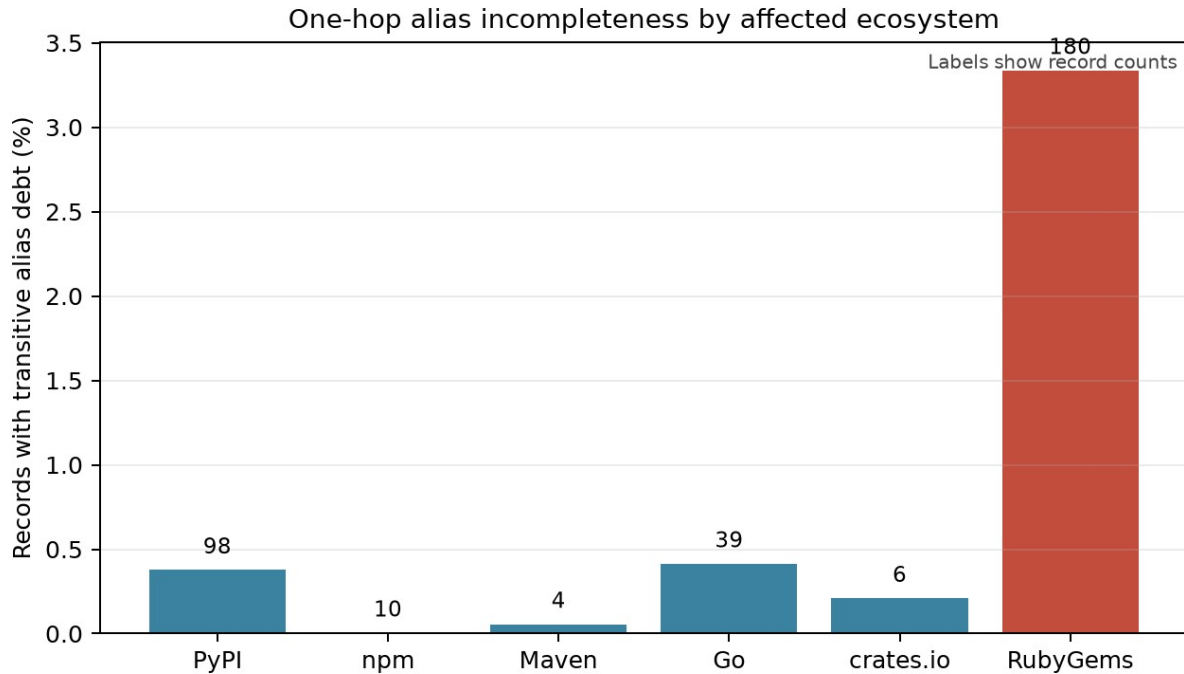


Figure 3. Empirical cumulative distribution of one-hop completeness among records with transitive alias debt.

Ecosystem stratification also showed concentration (Table 2 and Figure 2). RubyGems had the highest debt prevalence among the six prespecified ecosystems because the largest component consisted of records affecting distinct RubyGems packages. npm accounted for most records but only ten debt-bearing records.

Table 2. Transitive alias debt by affected ecosystem.

Ecosystem	Eligible records	Debt-bearing records	Debt prevalence
PyPI	25,995	98	0.3770%
npm	229,753	10	0.0044%
Maven	7,160	4	0.0559%
Go	9,421	39	0.4140%
crates.io	2,870	6	0.2091%
RubyGems	5,395	180	3.3364%

*Figure 2. Prevalence and counts of transitive alias debt in the six prespecified ecosystems.*

Counts by affected ecosystem are not mutually exclusive because a record may list packages in more than one ecosystem. Small additional ecosystems appearing inside records from the six archives are preserved in the machine-readable stratified table but are not interpreted because their counts are sparse.

4.3 Explicit reciprocity

There were 28,858 directed alias declarations whose targets were also record-backed IDs. Of these, 28,740 were explicitly reciprocal, a reciprocity rate of 99.591%. The remaining 118 declarations still formed undirected equality edges under the specification's symmetric semantics. A consumer that trusts the semantics and constructs undirected closure therefore resolves them; a consumer that requires the target record to repeat the source identifier may not.

This result separates semantic symmetry from explicit textual reciprocity. The specification tells consumers how to interpret aliases, but high explicit reciprocity reduces dependence on that inference and makes records easier to reconcile in systems that ingest sources incrementally.

4.4 Package breadth and publication span

Component package breadth was usually one, yet a small upper tail crossed package and ecosystem boundaries. The maximum package breadth was 406 and the maximum ecosystem breadth was ten. These observations do not prove incorrect aliasing. One vulnerability may affect multiple packages, and an advisory system may represent coordinated or bundled records differently. They do show why a merge operation should retain source records, relation provenance, and affected-package assertions rather than replacing a component with one flattened row.

All 10,986 multi-record components had at least two parseable publication timestamps. Their median publication span was 34.88 days, the 95th percentile was 1,615.73 days, and the maximum was 6,278.86 days. Publication timestamps belong to records, so the span may reflect source release schedules, later database ingestion, retrospective advisory creation, or corrected metadata. It should not be read as a vulnerability-disclosure delay.

4.5 Related-edge sensitivity

Adding related links changed the graph from 345,225 nodes and 267,681 components to 347,529 nodes and 267,405 components. Thus, 276 primary components were merged. The largest component increased from 361 to 1,556 nodes, a 4.31-fold expansion. Figure 4 compares these topological effects.

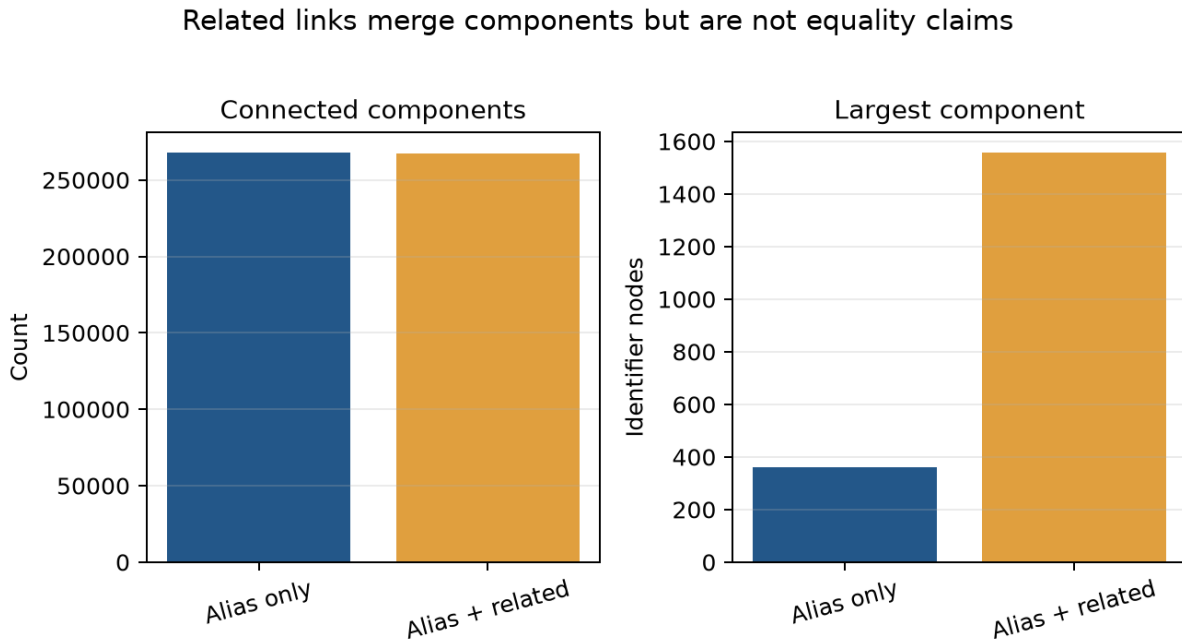


Figure 4. Component count and largest-component size with alias-only versus alias-plus-related edges.

The sensitivity result supports typed traversal. Related links may help analysts discover context, but treating them as equality would create much broader components than the alias relation alone. The case-folded package-name sensitivity left the headline package-breadth and multi-ecosystem counts unchanged. Excluding divergent IDs also made no difference because the six archives contained no cross-archive payload divergence after canonicalization.

5. Discussion

5.1 Main interpretation

The dominant result is not widespread fragmentation. Direct alias lists were sufficient for almost every record in this snapshot, and explicit record-to-record declarations were reciprocal more than 99.5% of the

time. That is a strong result for machine consumption. A tool that computes alias closure will resolve the remaining cases, but even a simpler one-hop implementation agrees with full closure for 99.881% of records overall.

The small residual matters for a different reason: averages hide graph concentration. More than half of all debt-bearing records belonged to one component. A resolver that encounters that component from any of its sparse record-level entry points can see only a small fraction of the identifiers without closure. The same corpus can therefore be excellent at population scale and still contain a few operationally awkward equivalence classes.

The findings also clarify what not to do. A system should not use the lower one-hop count as evidence that the source omitted a required field. Transitivity is an interpretation rule, so direct lists need not enumerate every reachable identifier. Nor should the system automatically flatten all component records into one unqualified advisory. A better representation keeps the component as a resolution layer while retaining each source record, package assertion, timestamp, and edge provenance.

5.2 Implications for vulnerability-management tools

A practical resolver can follow four rules.

First, store relationship type. Alias, upstream, and related links have different algebraic properties. Second, compute alias closure rather than issuing only a direct neighbor query. Union-find is sufficient for snapshot ingestion, while incremental systems can maintain components as records arrive. Third, preserve the path that established equivalence. Analysts should be able to see whether A was directly declared an alias of C or connected through B. Fourth, surface unusually broad components for review. Large package breadth, multiple ecosystems, or long publication spans are triage signals, not proof of error.

These rules also protect downstream metrics. Counting advisories without alias closure can overstate the number of distinct vulnerability identities. Flattening weak relations into aliases can understate it. Severity aggregation, ticket deduplication, and SBOM findings can all inherit those identity decisions.

5.3 Relation to prior work

Prior studies found meaningful inconsistencies in standardized vulnerability records [3–5] and showed that data-quality problems can distort later analyses [6]. This study finds a more favorable pattern for OSV alias declarations: explicit reciprocity is high and one-hop incompleteness is rare. The contribution is therefore not another general claim that vulnerability databases are unreliable. It identifies a bounded resolver behavior, measures its prevalence, and demonstrates why graph semantics still matter when the overall data are strong.

The result complements integrated knowledge-graph work [7]. Cross-database integration expands the evidence available to practitioners. The present findings suggest that such systems should keep relation types and closure provenance visible, especially when components contain many packages or when weak links produce large topological merges.

6. Threats to validity

6.1 Construct validity

Transitive alias debt measures the difference between direct lists and graph closure. It does not establish that the closure is factually correct. The analysis accepts the recorded alias relation as the primary equality claim because that is the semantics under study. Manual adjudication of 94,492 edges was outside scope.

Package breadth counts distinct normalized ecosystem-name pairs. Different ecosystems have different case rules, and aliases can legitimately span multiple packages. The case-folding sensitivity changed no headline breadth result, but package identity remains a descriptive construct.

Explicit reciprocity is not semantic symmetry. The OSV specification tells consumers to interpret aliases symmetrically [1]. Our reciprocity metric asks whether both record payloads independently state the relation when both records are available.

6.2 Internal validity

The pipeline parsed only fields required for the protocol and did not run the complete OSV JSON Schema validator over every record. ZIP CRC checks, parse accounting, duplicate-ID checks, graph invariants, unit tests, and deterministic reruns reduce implementation error. No divergent record IDs were observed across archives, so the prespecified variant-resolution rule did not affect outcomes.

The post hoc concentration diagnostics were added after the first run revealed the largest component. They are disclosed in a dated amendment and do not alter the primary corpus or estimates.

6.3 External validity

The results describe one snapshot acquired on 1 October 2026 and six package ecosystems. They do not establish temporal stability, behavior in operating-system distributions, or properties of records absent from these archives. The npm archive dominates the record count, so the corpus-wide prevalence is not an equal-weight ecosystem average. Ecosystem-specific results are therefore reported separately.

6.4 Operational validity

No production scanner, ticketing system, or remediation team was evaluated. The study does not quantify duplicated work, missed remediation, incident reduction, or monetary cost. Recommendations concern resolver architecture and reviewability, not measured organizational outcomes.

7. Reproducibility

The reproducibility package contains the frozen protocol and amendment, acquisition and analysis code, all 17 tests, dependency pins, source manifest and hashes, derived result tables, summary JSON, and figure-generation script. Raw official archives are omitted from the compact journal upload because of the portal's file-size limit, but their exact URLs, sizes, HTTP validators, and SHA-256 hashes are included. A full local archive retains the acquired snapshots.

The semantic-result SHA-256 is efc9a8c123ac03d322daf6f93d72f5b114255f8e2ea7e9d4a361e5af3c004186. A complete cached rerun reproduced this hash. Code written for the study is released under the MIT License. Original protocols, derived tables, figures, and documentation are released under CC BY 4.0. Upstream OSV records remain under their source terms.

Declarations

Funding: This research received no external funding.

Competing interests: A publisher-author relationship has been disclosed separately to the editorial office. Independent editorial handling and peer review are required. Identifying details are omitted here to preserve double-anonymous review.

Ethics: The study used public machine-readable vulnerability-advisory records and did not involve human participants, private personal data, live-system exploitation, or interaction with vulnerable systems. Institutional review-board approval was not required.

Data and code availability: The blinded reproducibility supplement accompanies the submission. It contains code, tests, exact source manifests, hashes, derived tables, figures, and rerun instructions. Official raw archives can be reacquired from the recorded OSV URLs; exact reproduction of the frozen snapshot requires the matching SHA-256 values.

Author contributions: The collective author group designed the study, reviewed the protocol, verified the data sources, audited the implementation and outputs, interpreted the results, and approved the submitted manuscript. The group accepts responsibility for the work.

Use of generative AI: Generative AI tools assisted with protocol development, code scaffolding, analysis execution, literature organization, and language drafting. The author group verified the sources, code, outputs, numerical claims, interpretations, and final text and accepts full responsibility. AI systems received no authorship or contributor credit.

References

- [1] Open Source Security Foundation. Open Source Vulnerability format: schema specification. Versioned source commit 65280faa676c7a69bb9208d876b2419f88dd7e03. 2026. <https://ossf.github.io/osv-schema/>
- [2] Open Source Vulnerabilities. Data sources and per-ecosystem data dumps. 2026. <https://google.github.io/osv.dev/data/>
- [3] Y. Dong, W. Guo, Y. Chen, X. Xing, Y. Zhang, and G. Wang. Towards the Detection of Inconsistencies in Public Security Vulnerability Reports. In: 28th USENIX Security Symposium, pp. 869–885, 2019. <https://www.usenix.org/conference/usenixsecurity19/presentation/dong>
- [4] A. Anwar, A. Abusnaina, S. Chen, F. Li, and D. Mohaisen. Cleaning the NVD: Comprehensive Quality Assessment, Improvements, and Analyses. *IEEE Transactions on Dependable and Secure Computing* 19(6), 4255–4269, 2022. <https://doi.org/10.1109/TDSC.2021.3125270>
- [5] Y. Jiang, M. Jeusfeld, and J. Ding. Evaluating the Data Inconsistency of Open-Source Vulnerability Repositories. In: *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021. <https://doi.org/10.1145/3465481.3470093>
- [6] R. Croft, M. A. Babar, and M. M. Kholoosi. Data Quality for Software Vulnerability Datasets. In: 45th IEEE/ACM International Conference on Software Engineering, 2023. <https://doi.org/10.1109/ICSE48619.2023.00022>
- [7] B. Boles, C. Izurieta, and A. M. Reinhold. Connecting the Dots: An Integrated Vulnerability Knowledge Graph for Security Practitioners. In: 2025 IEEE International Conference on Information Reuse and Integration and Data Science, 2025. <https://doi.org/10.1109/IRI66576.2025.00050>
- [8] J. Wunder, A. Corona, A. Hammer, and Z. Benenson. On NVD Users’ Attitudes, Experiences, Hopes, and Hurdles. *Digital Threats: Research and Practice* 5(3), 1–19, 2024. <https://doi.org/10.1145/3688806>