VULTRITION: Nutrition Labels for Vulnerability Datasets

Torge Hinrichs
Institute of Software Security
Hamburg University of Technology
Hamburg, Germany
torge.hinrichs@tuhh.de

Emanuele Iannone
Institute of Software Security
Hamburg University of Technology
Hamburg, Germany
emanuele.iannone@tuhh.de

Riccardo Scandariato
Institute of Software Security
Hamburg University of Technology
Hamburg, Germany
riccardo.scandariato@tuhh.de

***Abstract*—Function-level vulnerability detection models are often trained and tested on apparently equivalent datasets that differ in critical aspects such as class imbalance, the presence of duplicate functions, and contamination across splits. Such differences are often ill-documented and hard to verify.**

This paper introduces VULTRITION, a lightweight tool for analyzing function-level vulnerability datasets and generating compact summaries as *"nutrition labels"*. These labels report key facts about a dataset, such as its size, vulnerability-type coverage, class imbalance, redundancy, contamination across splits, and structural characteristics of the functions. These facts may help researchers make informed decisions when selecting datasets for vulnerability detection studies. We use VULTRITION to generate 15 nutrition labels for 9 popular function-level vulnerability datasets, revealing differences not readily apparent in the official dataset documentation. VULTRITION is planned to be extended to support other programming language types and vulnerability datasets beyond functions. We publicly release VULTRITION on GITHUB and as a PyPI package to support independent label generation and extensions. A video demo is available at: https://youtu.be/wEaGWKP5Szo.

***Index Terms*—Function-level vulnerability dataset, dataset nutrition labels, dataset quality assessment**

I. INTRODUCTION

Machine-learning-based vulnerability detection depends on labeled code datasets, typically at the function level, for training and evaluation. Model performance is heavily shaped by dataset quality: sample diversity, duplicates or cross-split contamination can produce deceptive results.

Recent approaches to vulnerability detection increasingly rely on learning-based models, including Transformer-based and large language models, to improve function-level detection [5], [1], [16]. Most of this work evaluates different methods on shared benchmarks, with Big-Vul being especially popular due to its size and coverage of many vulnerability categories [7]. Previous research has shown that vulnerability datasets often suffer from problems such as inconsistent metadata, duplicate samples, and train-test leakage, all of which can degrade the performance of models trained on them [4], [5].

In the last few years, the dataset landscape has expanded rapidly, with the recent introductions of TitanVul [16], ICVul [18], and SecVulEval [1]. Considering the datasets available to date, there is good coverage of programming languages, vulnerability types, and variety. While this diversity is valuable, it complicates dataset selection; simply choosing the newest or largest dataset does not guaranty that it is the most suitable benchmark for a particular study.

> ⚠ **Pain Point #1:** *Too many datasets to choose from.* Researchers lack guidance on selecting the right datasets for vulnerability detection studies.

Selecting a dataset requires more than checking its size or publication date. Datasets differ on many other critical aspects: class imbalance, share of duplicate functions, and contamination between train and test splits. While these *facts* appear relatively easy to compute, they are actually influenced by choices such as code similarity measures and the clone detection algorithm. However, dataset documentation, e.g., papers and README files, rarely report these facts in a standardized form, let alone describe exactly how they were measured. Consequently, researchers must recompute these facts using custom heuristics and assumptions, thereby negatively affecting comparability across studies.

> ⚠ **Pain Point #2:** *No standard way to extract facts.* Researchers must adopt custom heuristics and assumptions to extract facts from vulnerability datasets.

A further complication is that datasets continue to change after their initial release. Maintainers may add or remove entries, fix labeling errors, or adjust preprocessing pipelines. This leads to new versions, while the original paper remains the primary reference for researchers. Thus, two studies may formally cite the same dataset paper but actually use two different versions of it, hindering independent replications. For example, the original release of PrimeVul contains 235,768 functions, of which 6,968 are vulnerable [5], while the updated v0.1 has 224,533 functions, with 6,004 labeled as vulnerable. In total, ∼8,000 functions were removed.

> ⚠ **Pain Point #3:** *Datasets evolve, but citations do not.* Researchers cannot reliably reproduce a vulnerability detection study when the dataset version changes.

To alleviate these pain points, we introduce VULTRITION, a lightweight tool that analyzes a function-level vulnerability dataset and generates a **nutrition label**. This label reports relevant *facts* about the dataset, such as the number of entries (functions), metadata completeness, vulnerability-type cover-

age, class imbalance, proportion of duplicate functions, level of contamination across splits, and structural characteristics of the functions (e.g., the number of tokens). These labels are similar to nutrition facts on food products, listing quantities of fat, carbohydrates, salt, and other components. Although they do not directly measure a dataset's intrinsic "quality," they provide clear visual information that helps researchers (i) choose suitable datasets for their studies and (ii) interpret benchmark results more accurately.

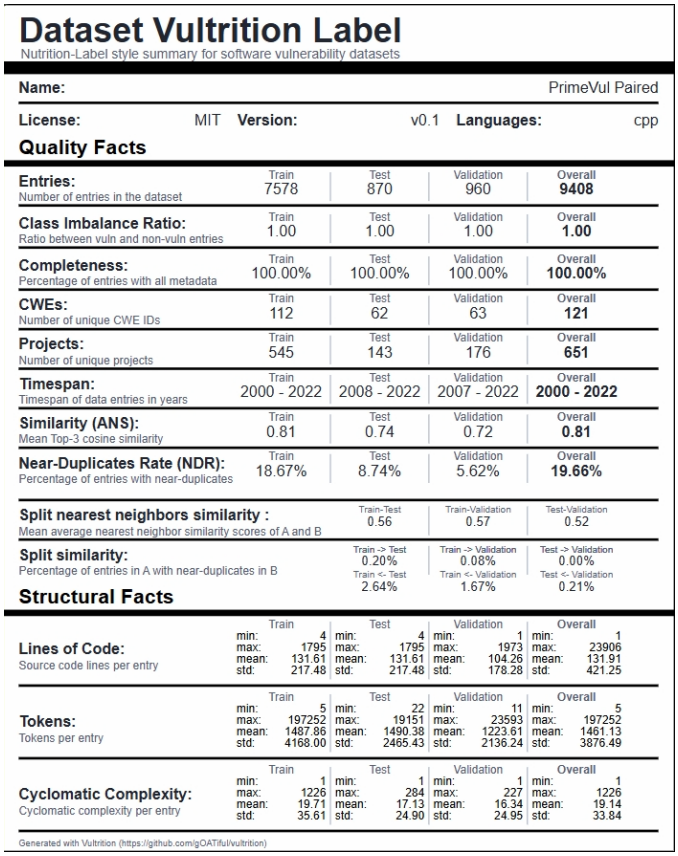
We demonstrate the usefulness of VULTRITION through a comparative case study involving 9 popular function-level vulnerability datasets across 4 programming languages (C, C++, Java, and Python), resulting in 15 nutrition labels. The study revealed findings not readily apparent from the datasets' documentation, such as substantial differences in within-dataset redundancy and significant cross-contamination between predefined evaluation splits, potentially leading to data leakage. VULTRITION is publicly available on GITHUB(https://github.com/tuhh-softsec/vultrition) and as a PyPI package(https://pypi.org/project/vultrition/) for independent label generation, also the labels we created for the datasets can be found there.

II. RELATED WORK

Prior work has shown that vulnerability datasets significantly affect the reliability of benchmarking. Croft et al. [4] identified several quality problems in them, mainly inaccurate labels and duplicate entries, that directly affect model performance. Esposito and Falessi [6] and Guo et al. [9] further surveyed vulnerability datasets, highlighting challenges around availability, documentation, reuse, and desirable properties.

A second line of work proposed standardized documentation for machine-learning artifacts. Datasheets for datasets [8] and model cards [19], particularly those adopted by Hugging Face [12], established that datasets (and models) should be released with clear information about composition, evaluation conditions, and intended use. However, such datasheets mostly consist of long natural-language text, which may be difficult to read and interpret, whereas the cards do not systematically report numerical facts about dataset quality.

In this paper, we pursue the same goal by proposing a complementary view of sheets and cards that reports key numerical facts in a compact format. We call these summaries *"nutrition labels"*, drawing inspiration from food package labels, which report only numerical facts without direct qualitative judgment or high-level analyses. This differs from the existing "Dataset Nutrition Label" proposed by Holland et al. [11] and refined by Chmielinski et al. [3], which contains a mix of natural language text and plots. Stoyanovich and Howe [21] proposed other data nutrition labels but demonstrated their use only for rankings (e.g., of university departments). Kelley et al. [14] also proposed nutrition labels for privacy policies rather than datasets. VULTRITION specifically targets **function-level vulnerability datasets**, motivated by two key gaps: (1) their rapid proliferation in recent years, and (2) the absence of any tool that systematically computes relevant facts about them.



Dataset Vultrition Label

Nutrition-Label style summary for software vulnerability datasets

Name: PrimeVul Paired

License: MIT **Version:** v0.1 **Languages:** cpp

Quality Facts

	Train	Test	Validation	Overall
Entries: Number of entries in the dataset	7578	870	960	**9408**
Class Imbalance Ratio: Ratio between vuln and non-vuln entries	1.00	1.00	1.00	**1.00**
Completeness: Percentage of entries with all metadata	100.00%	100.00%	100.00%	**100.00%**
CWEs: Number of unique CWE IDs	112	62	63	**121**
Projects: Number of unique projects	545	143	176	**651**
Timespan: Timespan of data entries in years	2000 - 2022	2008 - 2022	2007 - 2022	**2000 - 2022**
Similarity (ANS): Mean Top-3 cosine similarity	0.81	0.74	0.72	**0.81**
Near-Duplicates Rate (NDR): Percentage of entries with near-duplicates	18.67%	8.74%	5.62%	**19.66%**

	Train-Test	Train-Validation	Test-Validation
Split nearest neighbors similarity : Mean average nearest neighbor similarity scores of A and B	0.56	0.57	0.52

	Train -> Test	Train -> Validation	Test -> Validation
Split similarity: Percentage of entries in A with near-duplicates in B	0.20%	0.08%	0.00%
	Train <- Test	Train <- Validation	Test <- Validation
	2.64%	1.67%	0.21%

Structural Facts

	Train	Test	Validation	Overall
Lines of Code: Source code lines per entry	min: 4 max: 1795 mean: 131.61 std: 217.48	min: 4 max: 1795 mean: 131.61 std: 217.48	min: 1 max: 1973 mean: 104.26 std: 178.28	min: 1 max: 23906 mean: 131.91 std: 421.25
Tokens: Tokens per entry	min: 5 max: 197252 mean: 1487.86 std: 4168.00	min: 22 max: 19151 mean: 1490.38 std: 2465.43	min: 11 max: 23593 mean: 1223.61 std: 2136.24	min: 5 max: 197252 mean: 1461.13 std: 3876.49
Cyclomatic Complexity: Cyclomatic complexity per entry	min: 1 max: 1226 mean: 19.71 std: 35.61	min: 1 max: 284 mean: 17.13 std: 24.90	min: 1 max: 227 mean: 16.34 std: 24.95	min: 1 max: 1226 mean: 19.14 std: 33.84

Generated with Vultrition (https://github.com/gOATiful/vultrition)

Fig. 1. Nutrition label of PRIMEVUL PAIRED generated by VULTRITION. The facts about the splits are shown as they are predefined in this dataset.

III. DESIGNING VULTRITION AND NUTRITION LABELS

A. Dataset Nutrition Facts

VULTRITION computes **14 facts** in total, grouped into two categories: *quality* and *structural*. When the input dataset includes predefined splits (e.g., *train*, *validation*, *test*), VULTRITION additionally computes all facts per split and conducts a cross-contamination analysis. An example of a label is shown in Figure 1.

1) Quality Facts: These facts capture properties that concern the dataset's completeness, coverage, and reliability. VULTRITION counts the **number of functions** and the percentage of **complete functions** with all essential data fields (source code, label, CVE, CWEs, and project name). VULTRITION also measures **class imbalance** as the ratio of vulnerable to non-vulnerable functions. VULTRITION characterizes *coverage* through (1) the number of **unique projects**, (2) **unique CWEs**, and (3) the **timespan** of the functions' CVEs in terms of *years*.

VULTRITION also characterizes the *redundancy* of functions in the dataset. First, it creates dense embeddings of each function with JINA-CODE-EMBEDDINGS, an embedding model specialized for code retrieval [15]. Then, it creates a search index with FAISS, a library for efficient similarity search of dense vectors [13], and, for each function, retrieves its three

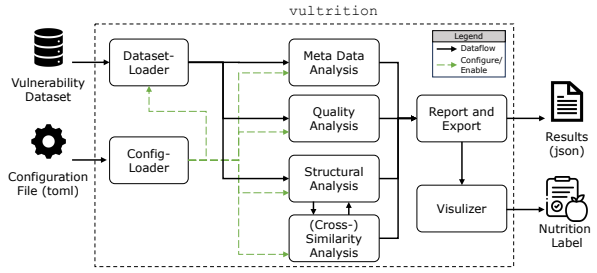


Fig. 2. Overview of the VULTRITION pipeline.

nearest neighbors (i.e., the most similar functions excluding itself) according to cosine similarity. VULTRITION reports two groups of metrics. First, the **average of the 3 nearest-neighbor similarity (ANS)**, which acts as a broad, objective measure of similarity across all functions. We chose the three most similar functions rather than one to account for the fact that some datasets are generated from *vulnerability-fixing commits*. These datasets consist of pre- and post-fix versions of the same functions, which often differ only slightly due to the patch. This makes the 1-nearest neighbor likely to have a similarity score close to 1, especially for larger functions. Second, the **near-duplicate rate (NDR)**, defined as the fraction of functions whose 3-nearest-neighbor similarity exceeds a configurable threshold (defaulted to 0.95). This offers a directly interpretable value on overall redundancy.

If the dataset has predefined splits, VULTRITION also assesses *cross-contamination* among them using. For each pair of splits $\langle A, B\rangle$, VULTRITION first creates the embeddings of all functions in A and B and then creates two FAISS indexes, one for A and one for B. For each function in A, VULTRITION retrieves its 3-nearest-neighbors in B according to cosine similarity; the same procedure is repeated the other way around (functions in B are mapped to the 3-nearest-neighbors in A). VULTRITION produces three facts for each pair $\langle A, B\rangle$: (1) the **harmonic mean of ANS**(A) **and ANS**(B) **scores**; (2) the **percentage of functions in** A **that have near-duplicates in** B (i.e., NDR(A, B)); (3) same as (2) but in the other direction (i.e., NDR(B, A)).

2) Structural Facts: These facts describe the length and complexity of individual functions. Namely, VULTRITION computes the minimum, maximum, mean, and standard deviation of **lines of code**, **token count**, and **cyclomatic complexity**. Lines of code and cyclomatic complexity are obtained with `lizard`. Token count, computed with `tiktoken` and can be configured, has become relevant in today's research, as functions with many tokens that do not fit in LLMs' context windows lead to incorrect training and testing.

B. How to Run VULTRITION

VULTRITION is a *command-line* tool that generates facts about a function-level vulnerability dataset and a graphical nutrition label. Its input consists of the dataset to analyze (supporting CSV, JSON, and JSONL formats) and a configuration file (TOML format). Its working can be seen in Figure 2.

TABLE I
SELECTED VULNERABILITY DATASETS OF LABELED FUNCTIONS.

Name	Authors	Release Year	Variants	Train/Valid/Test
SecVulEval [1]	Ahmed et al.	2025	-	✗
ICVul [18]	Lu et al.	2025	-	✗
TitanVul [16]	Li et al.	2025	-	✗
MegaVul [20]	Ni et al.	2024	C++	✗
			Java	✗
ReposVul [22]	Wang et al.	2024	Full	✓
			C	✗
			C++	✗
			Java	✗
			Python	✗
PrimeVul [5]	Ding et al.	2024	Full	✓
			Paired	✓
CleanVul [17]	Li et al.	2024	-	✗
DiverseVul [2]	Chen et al.	2023	-	✗
Big-Vul [7]	Fan et al.	2020	-	✗

The user must manually fill the configuration file with (i) *metadata* information about the dataset (name, license, and version) and (ii) the name of the fields/columns of the dataset file from which VULTRITION can retrieve the source code, label, CVE, CWEs, and project name of each function (further details are in VULTRITION's repository [10]). To simplify this step, VULTRITION can be run with a special flag to produce a template configuration file ready to be filled in. VULTRITION returns the dataset facts in a simple JSON file (an example is in the tool's repository [10]). The JSON file is then converted into a graphical nutrition label in SVG format using `svgwrite`.

IV. CASE STUDY

We used VULTRITION to generate nutrition labels for a set of well-known function-level vulnerability datasets. This case study aims to show how labels can easily reveal interesting differences across datasets and facilitate comparisons; thus, the goal was not to judge dataset quality or to rank them.

A. Dataset Selection

We first selected a collection of widely used and emerging datasets containing labeled vulnerable and non-vulnerable functions. The selection was mainly driven by the author's knowledge of the topic and queries on GOOGLE SCHOLAR. Candidate datasets were required to (1) include functions' source code, (2) be publicly available, and (3) be provided in a machine-processable format (e.g., CSV, JSON, JSONL). We excluded datasets that contained only CVEs or vulnerability-fixing commits to avoid imposing our own function-extraction and labeling procedures; instead, we used existing datasets as-is, mirroring typical consumer use.

We selected **9 datasets** in total, summarized in Table I. This set is illustrative rather than comprehensive and is intended to showcase the application of nutrition labels. The datasets differ in size, language coverage, construction methods, and available metadata, making them useful for evaluating whether the labels correctly capture these properties. Some datasets, i.e. MegaVul, ReposVul, and PrimeVul define additional "sub-datasets." We treat each of these subsets as a separate dataset, yielding **15 nutrition labels** in total.

TABLE II
A SELECTION OF FACTS FROM THE NUTRITION LABELS GENERATED BY VULTRITION ON 15 DATASETS.

Dataset	Samples	CIR	#Projects	#CWEs	ANS	NDR (%)	Lines Of Code				Cyclomatic Complexity				Tokens			
							Min	Max	Mean	Std	Min	Max	Mean	Std	Min	Max	Mean	Std
SecVulEval	25,440	0.76	707	145	0.84	17.20	1	6,291	75.94	156.08	1	1,226	12.40	23.04	6	52,983	853.13	1,793.38
ICVul	15,396	0.69	677	136	0.82	13.35	1	3,510	80.07	158.15	1	545	13.24	22.45	5	52,983	927.62	2,036.22
TitanVul	77,096	1.00	5,384	254	0.85	22.90	1	23,906	61.95	175.88	1	1,226	11.26	18.35	4	197,252	667.73	1,640.59
MegaVul-C++	353,873	0.05	1,062	176	0.84	13.78	1	759	22.23	30.47	1	478	4.85	6.94	3	28,321	240.44	366.63
MegaVul-Java	41,949	0.06	362	115	0.76	5.39	1	678	12.50	15.94	1	156	3.14	4.14	3	8,117	122.58	178.11
ReposVul-Full*	232,239	0.03	601	152	0.90	43.45	1	6,312	29.72	66.92	1	1,192	5.93	11.31	2	65,102	323.26	828.54
ReposVul-C++	20,460	0.00	85	58	0.92	43.41	1	3,338	23.99	64.77	1	740	4.50	11.52	2	35,023	295.57	977.64
ReposVul-C	181,694	0.01	325	126	0.94	57.53	1	6,312	30.89	77.52	1	1,192	6.00	12.46	3	65,102	332.24	933.20
ReposVul-Java	3,042	0.06	204	96	0.83	16.63	1	246	12.24	18.78	1	63	2.35	3.92	19	115,272	3,029.85	7,247.54
ReposVul-Python	27,269	0.01	252	124	0.89	41.62	2	3,892	23.99	158.05	1	3	1.16	0.45	5	107,194	428.68	1,591.15
PrimeVul-Full*	224,533	0.03	751	130	0.83	9.98	1	23,906	34.49	114.92	1	1,226	6.34	14.12	1	197,252	370.62	1,175.08
PrimeVul-Paired*	9,408	1.00	651	121	0.81	19.66	1	23,906	131.91	421.25	1	1,226	19.14	33.84	5	197,252	1,461.13	3,876.49
CleanVul	12,102	1.00	1,473	76	0.80	12.74	1	531	53.86	63.32	1	183	11.52	14.91	7	5,586	562.80	678.85
DiverseVul	330,491	0.06	800	150	0.85	16.36	1	23,906	34.91	108.93	1	1,226	6.32	12.54	1	197,252	379.02	1,136.43
BigVul	188,636	0.06	310	91	0.83	10.82	1	6434	24.59	61.00	1	1,225	4.83	9.81	4	45,205	254.30	657.03

*have predefined train/validation/test splits; CIR = Class Imbalance Ratio; ANS = Average Top-3-Nearest-neighbor Similarity; NDR = Near-Duplicate Rate (0.95 threshold).

Furthermore, ReposVul (only the "Full" variant) and both PrimeVul variants also come with predefined splits into train, validation, and test sets decided by their authors. In such cases, VULTRITION also computed all facts for each of the splits and ran the cross-contamination analysis. These additional facts are incorporated into the labels of the belonging dataset.

B. Key Findings

Table II summarizes key facts from the 15 nutrition labels that VULTRITION generated. We report the most interesting findings. First, the 3-nearest-neighbor analysis reveals substantial within-dataset redundancy, although its extent varies considerably across datasets. ReposVul-Full exhibits the highest values, with an ANS of 0.90 and an NDR of 43%. The other variants of ReposVul exhibit high NDR (more than 40%), except for the Java variant, which drops to 16%. TitanVul and DiverseVul also exhibit a high degree of redundancy, with the same ANS score of 0.85 and NDRs of 23% and 16%, respectively. While PrimeVul-Paired has a lower ANS than other datasets, it still has 20% NDR. By contrast, PrimeVul-Full, BigVul, and CleanVul have lower NDR values, ranging from 10% to 13%. Interestingly, although CleanVul and PrimeVul-Paired are built from fixed function pairs, their ANS values are only 0.80 and 0.81, respectively. This suggests that a function's 3-neighborhood is very likely to contain its "counterpart," but the other two neighbors are less similar.

Second, the datasets vary notably in class balance and code complexity. CleanVul, PrimeVul-Paired, and TitanVul are perfectly balanced (1.00), i.e., containing equal numbers of vulnerable and non-vulnerable functions. In contrast, BigVul, DiverseVul, MegaVul, PrimeVul-Full, and ReposVul-Full are heavily skewed toward non-vulnerable functions. In terms of function size, PrimeVul-Paired features the longest functions on average, with $\sim$132 LOC and $\sim$1461 tokens. ICVul and SecVulEval also include relatively large functions, with a mean LOC of $\sim$80 ($\sim$928 tokens) and $\sim$76 ($\sim$853 tokens), respectively, while BigVul and MegaVul consist of considerably shorter functions on average. This fact is important because ML models operate under constrained context windows and might not fit these functions without preprocessing.

Finally, the cross-contamination analysis reveals a particularly high risk in the predefined splits of ReposVul. 28% of test functions and 28% of validation functions have a near-duplicate in the training split. This behavior differs from the predefined PrimeVul splits: the corresponding near-duplicate rates are at most 2% for PrimeVul-Full and 3% for PrimeVul-Paired. Consequently, evaluations performed on predefined ReposVul splits may be substantially affected by test or validation data leaked into the training data.

V. DISCUSSION

A. The Importance of Nutrition Labels

Dataset nutrition labels help **researchers** select the right dataset for their function-level vulnerability detection studies based on transparent metrics (*Pain Point #1*). They also help them better contextualize model performance across multiple datasets, reducing the risk of wrong conclusions. VULTRITION also comes with a catalog of 15 pre-generated labels for the datasets used in our case study (Section IV) that anyone can reuse. Still, researchers can reuse VULTRITION to generate additional labels for more datasets or for custom variants.

Nutrition labels also help **dataset curators** determine a clear release format to document the version, scope, and limitations of their artifacts (*Pain Point #2 & #3*). In particular, the labels are designed to be integrated into the dataset's README file to serve as a visual summary, similar to Hugging Face dataset cards.

Furthermore, these labels may enable **reviewers** to quickly examine the dataset(s) reported in a paper, helping them spot missing metadata, possible contamination, or claims lacking numerical evidence.

B. Current Limitations and Validation Plan

VULTRITION and its nutrition labels are designed to **increase transparency** rather than to judge or rank the overall quality of a dataset. They can pinpoint potential issues such as duplicate entries or cross-set contamination, but they do not replace the need for manual inspection. The metrics currently implemented in VULTRITION are grounded in the analysis by Croft et al. [4] and have been updated to leverage modern techniques, such as code similarity search in a dense vector space. However, the selected metrics do not exhaustively cover all aspects of dataset quality. To address this, we plan to **systematically survey the literature** to compile a comprehensive

inventory of candidate metrics, and to **interview researchers** on which metrics best capture dataset quality in practice.

At the same time, some metrics currently implemented in VULTRITION may not align with users' interpretations. In particular, the uniqueness and cross-contamination facts are sensitive to the method and threshold we chose: two functions are considered near-duplicates when their similarity exceeds 0.95. Thus, we plan to conduct an **empirical study to validate these metrics**, comparing different measurement methods and collecting human opinions on which approach best reflects the intended notion of similarity.

After defining the most suitable metric suite, we will assess the **practical usefulness of dataset nutrition labels** via a user study. We will examine whether these labels (i) enable users to choose vulnerability datasets more efficiently than without such guidance and (ii) are regarded as *useful*.

VI. Concluding Remarks

Benchmark performance alone is insufficient to assess the reliability of vulnerability detection models. To address this gap, we introduced dataset nutrition labels and VULTRITION to guide the selection of datasets for function-level vulnerability detection studies. While the labels do not certify dataset quality, they increase transparency and allow researchers to make more informed decisions.

In addition, we have identified three main actions for future development. First, we plan to integrate automated checks that verify (i) syntactic correctness of functions, (ii) whether functions labeled as vulnerable are really vulnerable, and (iii) whether assigned CWEs are accurate. Second, we will enable VULTRITION to generate nutrition labels for datasets of other software engineering domains, such as bug and code-smell detection. Finally, we will establish a publicly accessible catalog of nutrition labels that the community can collectively maintain and expand.

Acknowledgments

This work was partially supported by EU-funded project Sec4AI4Sec (grant no. 101120393). AI tools (ChatGPT and ElevenLabs) were used to support the creation of the video demonstration and streamline the source code of VULTRITION. The authors reviewed and take responsibility for the final content.

References

[1] Md Basim Uddin Ahmed, Nima Shiri Harzevili, Jiho Shin, Hung Viet Pham, and Song Wang. Secvuleval: Benchmarking llms for real-world c/c++ vulnerability detection. *arXiv preprint arXiv:2505.19828*, 2025.

[2] Yizheng Chen, Zhoujie Ding, Lamya Alowain, Xinyun Chen, and David Wagner. Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 654–668, 2023.

[3] Kasia S. Chmielinski, Sarah Newman, Matt Taylor, Josh Joseph, Kemi Thomas, Jessica Yurkofsky, and Yue Chelsea Qiu. The dataset nutrition label (2nd gen): Leveraging context to mitigate harms in artificial intelligence, 2022.

[4] Roland Croft, Muhammad Ali Babar, and M. Mehdi Kholoosi. Data quality for software vulnerability datasets. In *Proceedings of the 45th International Conference on Software Engineering*, ICSE '23, pages 121–133. IEEE Press, 2023.

[5] Yangruibo Ding, Yanjun Fu, Omniyyah Ibrahim, Chawin Sitawarin, Xinyun Chen, Basel Alomair, David Wagner, Baishakhi Ray, and Yizheng Chen. Vulnerability detection with code language models: How far are we? *arXiv preprint arXiv:2403.18624*, 2024.

[6] Matteo Esposito and Davide Falessi. Validate: A deep dive into vulnerability prediction datasets. *Information and Software Technology*, 170:107448, 2024.

[7] Jiahao Fan, Yi Li, Shaohua Wang, and Tien N. Nguyen. A c/c++ code vulnerability dataset with code changes and cve summaries. In *Proceedings of the 17th International Conference on Mining Software Repositories*, pages 508–512, 2020.

[8] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, and Kate Crawford. Datasheets for datasets. *Communications of the ACM*, 64(12):86–92, 2021.

[9] Yuejun Guo, Seifeddine Bettaieb, and Fran Casino. A comprehensive analysis on software vulnerability detection datasets: trends, challenges, and road ahead. *International Journal of Information Security*, 23(5):3311–3327, 2024.

[10] Torge Hinrichs, Emanuele Iannone, and Riccardo Scandariato. VULTRITION. https://github.com/tuhh-softsec/vultrition, 2026. Accessed: 2026-05-26.

[11] Sarah Holland, Ahmed Hosny, Sarah Newman, Joshua Joseph, and Kasia Chmielinski. The dataset nutrition label: A framework to drive higher data quality standards, 2018.

[12] Hugging Face. Dataset cards. https://huggingface.co/docs/hub/datasets-cards, 2026. Accessed: 2026-05-28.

[13] Jeff Johnson, Matthijs Douze, and Herve Jegou. Billion-Scale Similarity Search with GPUs . *IEEE Transactions on Big Data*, 7(03):535–547, July 2021.

[14] Patrick Gage Kelley, Joanna Bresee, Lorrie Faith Cranor, and Robert W. Reeder. A "nutrition label" for privacy. In *Proceedings of the 5th Symposium on Usable Privacy and Security (SOUPS 2009)*, 2009.

[15] Daria Kryvosheieva, Saba Sturua, Michael Günther, Scott Martens, and Han Xiao. Efficient code embeddings from code generation models, 2025.

[16] Yikun Li, Ngoc Tan Bui, Ting Zhang, Chengran Yang, Xin Zhou, Martin Weyssow, Jinfeng Jiang, Junkai Chen, Huihui Huang, Huu Hung Nguyen, Chiok Yew Ho, Jie Tan, Ruiyin Li, Yide Yin, Han Wei Ang, Frank Liauw, Eng Lieh Ouh, Lwin Khin Shar, and David Lo. Out of distribution, out of luck: How well can llms trained on vulnerability datasets detect top 25 cwe weaknesses? *arXiv preprint arXiv:2507.21817*, 2025.

[17] Yikun Li, Ting Zhang, Ratnadira Widyasari, Yan Naing Tun, Huu Hung Nguyen, Tan Bui, Ivana Clairine Irsan, Yiran Cheng, Xiang Lan, Han Wei Ang, Frank Liauw, Martin Weyssow, Hong Jin Kang, Eng Lieh Ouh, Lwin Khin Shar, and David Lo. Cleanvul: Automatic function-level vulnerability detection in code commits using llm heuristics. *arXiv preprint arXiv:2411.17274*, 2024.

[18] Chaomeng Lu, Tianyu Li, Toon Dehaene, and Bert Lagaisse. Icvul: A well-labeled c/c++ vulnerability dataset with comprehensive metadata and vccs. In *Proceedings of the 22nd IEEE/ACM International Conference on Mining Software Repositories*, pages 154–158, 2025.

[19] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19)*, pages 220–229, 2019.

[20] Chao Ni, Liyu Shen, Xiaohu Yang, Yan Zhu, and Shaohua Wang. Megavul: A c/c++ vulnerability dataset with comprehensive code representation. In *Proceedings of the 21st International Conference on Mining Software Repositories*, pages 738–742, 2024.

[21] Julia Stoyanovich and Bill Howe. Nutritional labels for data and models. *IEEE Data Engineering Bulletin*, 42(3):13–23, 2019.

[22] Xinchen Wang, Ruida Hu, Cuiyun Gao, Xin-Cheng Wen, Yujia Chen, and Qing Liao. Reposvul: A repository-level high-quality vulnerability dataset. In *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pages 472–483, 2024.