Poster: Software Vulnerability Detection: A Grand Challenge for AI

1st Aayush Gupta
Computer Science
University of Houston
Houston, Texas, United States
agupta56@cougarnet.uh.edu

2nd Arthur Dunbar
Computer Science
University of Houston
Houston, Texas, United States
apdunbar@cougarnet.uh.edu

3rd Rakesh M. Verma*
Computer Science
University of Houston
Houston, Texas, United States
rmverma2@central.uh.edu

***Abstract*—Bugs in software are notorious for causing rocket crashes, equipment failures, and enabling attacks on critical infrastructure. Fuzzing and manual detection are the best approaches so far, both of which are incomplete and cannot prove the absence of bugs. Machine learning approaches to automatic bug detection have been studied since the late 1990s, but their record is mixed at best. Large language models have also struggled at this task. In this poster, we propose automatic bug detection as a grand challenge for artificial intelligence and suggest a way forward which includes an approach based on multimodal models, quality datasets, and customized prompts.**

***Index Terms*—software bug detection, multimodal models, prompt engineering**

I. Grand Challenge

Bugs in software have caused enormous damage especially in the interconnected age of the internet.[1] Fuzzing [16] and manual detection (aka debugging) are the approaches to find and fix bugs. Both are known to be incomplete. Run-time verification [11] and combinations of static with dynamic analysis have also been proposed. Automatic detection of bugs using machine learning has been receiving increasing attention since at least 2018 [4], [7]. However, machine learning models including large language models have struggled with this task and the problem of software vulnerability detection is as prevalent as ever. Hence, we propose the task of automatic software vulnerability detection as a grand challenge for AI.

To suggest a way forward, we first examine nine popular datasets for this task and reveal their shortcomings. We then propose our vision for this challenge consisting of multimodal models, high quality datasets and customized prompts.

II. Datasets

Numerous datasets have been proposed to address the challenges of vulnerability detection. Here, we focus on the following nine datasets because of their size, popularity, and machine learning suitability:

1) Big-Vul [5] gathers vulnerability fixes from GitHub projects with CVE metadata. 3,754 bugs 91 types.
2) Devign [15] manually labeled datasets from four open-source C projects (2 released publicly).
3) DiverseVul [3] 18,945 vulnerable functions covering 150 CWEs and 330,492 non-vulnerable functions from diverse projects.
4) MegaVul [9] expands on Big-Vul making a larger, more diverse dataset with 17,380 vulnerabilities of 169 types.
5) RealVul [1] tries to simulate real-world scenarios and include entire codebases.
6) MVDSC [14] normalizes variable and function names for generalization.
7) VulDeePecker (VDP) [8] uses code gadgets to represent programs.
8) ReVeal [2] focused on the limitations of current datasets. Emphasizes realistic settings.
9) D2A [13] sources data from GitHub. Makes use of static analysis on before and after of bug fixes to reduce errors.

III. Results

We evaluate the software vulnerability datasets using four metrics we created. These four metrics can be found in Table I,II,III and on the classifier difficulty score table found on our poster. Each of these four metrics generates a score for each dataset, which is scaled so that the largest score is a 1. These scores are then averaged together to give an overall score for the dataset, which can be found in Table IV. We analyze the performance of the aforementioned machine learning models across these datasets. We also analyze the results of running the datasets through CleanLab [10], what the datasets return for Vendi Scores [6] based upon N-grams of 1 to 3, and we also score the datasets based on their coverage of code constructs.

IV. Proposed Roadmap

We suggest that: (a) high quality datasets should be constructed that retain the context necessary for the manifestation of their bugs, (b) usage of multiple modalities, e.g., abstract syntax tree, code property graphs [12], and the code itself, be provided to either an ensemble of models or a multimodal model. With such an approach the model has the potential to pick up true signals of bugs, and (c) prompt engineering in case LLMs are used. To measure data quality, we use these metrics, ground truth, coverage and diversity. Ground truth means that the non-vulnerable examples should be chosen carefully so as not to contain any vulnerabilities. By coverage we mean, for

*On Sabbatical during AY 2024-25 at NSWC Dahlgren, Virginia

[1]https://www.pingdom.com/blog/10-historical-software-bugs-with-extreme-consequences/-accessed14April2025.

TABLE I

CLEANLAB RESULTS. THESE ARE RESULTS FROM RUNNING CLEANLAB (* 50,000 SAMPLED BECAUSE REALVUL'S EXAMPLES ARE TOO BIG TOO HOLD IN MEMORY). THE CLEANLAB RESULTS SCORE (CRS) COLUMN IS DETERMINED BY (1 - MAX(0, (NEAR DUP%-.2)*1.25) + 1 - LABEL INC% + 1 - (OUTLIERS% * 10) + 1 / (1 + UNDERPERFORMING GROUPS (UPG))) / 4.

Dataset	ND	LI	Outlier	UPG	Score
ReVeal	19.15%	3.69%	0.47%	0	0.992
Devign	12.89%	33.57%	0.93%	0	0.905
MVDSC	10.07%	5.68%	4.60%	7	0.661
VDP	60.21%	25.72%	5.97%	6	0.452
RealVul*	7.53%	0.07%	2.65%	0	0.946
BigVul	4.20%	3.47%	0.18%	0	1.000
MegaVul	0.64%	2.25%	0.49%	0	0.993
Diversevul	18.34%	2.57%	0.38%	5	0.786
D2A	99.48%	1.00%	8.37%	7	0.327

TABLE II

VENDI SCORE RESULTS. THE VENDI SCORE (VS) IS DETERMINED BY SOFTMAXING THE LOG OF THE VENDI SCORE. THE N-GRAMS ARE SIZE 1 TO 3.

Dataset	N-gram VS	Score
ReVeal	1263.95	0.511
Devign	916.91	0.371
MVDSC	81.03	0.033
VDP	87.50	0.035
RealVul	1149.98	0.465
BigVul	2471.78	1.000
MegaVul	1015.81	0.411
Diversevul	2297.56	0.930
D2A	124.06	0.050

TABLE III

CONSTRUCTS COVERAGE RESULTS. CONSTRUCT COVERAGE IS THE NUMBER OF PROGRAMMING CONSTRUCTS COVERED BY AT LEAST ONE EXAMPLE IN THE DATASET. THE CONSTRUCT COVERAGE SCORE (CCS) IS DETERMINED BY PERCENT OF ESTIMATED AVAILABLE CONSTRUCTS GIVEN THE LANGUAGES USED.

Dataset	has c++?	CS	Score
ReVeal	yes	260	0.729
MVDSC	yes	163	0.457
VDP	yes	189	0.530
RealVul	yes	338	0.948
BigVul	yes	291	0.816
MegaVul	yes	315	0.884
Diversevul	yes	321	0.900
D2A	yes	229	0.642
C++ Total Est.	yes	416	-
Devign	no	234	1.000
C Total Est.	no	275	-
All Datasets Coverage	yes	346	-

the set of programming languages used in the dataset, how many of the code constructs that exist in those languages are represented in the dataset. By diversity we seek to measure also the different types of bugs and the different ways in which they are manifested in the vulnerable examples.

ACKNOWLEDGMENTS

Research partly supported by NSF grants 2210198 and 2244279, ARO grant W911NF-23-1-0191, and an ONR Sabbatical Faculty Fellowship. Verma is the founder of Everest Cyber Security and Analytics, Inc.

TABLE IV

TOTAL SCORE (TS) CALCULATED BY AVERAGING THE OTHER SCORES. CDS OR CLASSIFIER DIFFICULTY SCORE CAN BE FOUND ON OUR POSTER

Dataset	CDS	CRS	VS	CCS	TS
ReVeal	0.960	0.992	0.511	0.729	0.798
Devign	0.671	0.905	0.371	1.000	0.737
MVDSC	0.249	0.661	0.033	0.457	0.350
VDP	0.260	0.452	0.035	0.530	0.319
RealVul	0.552	0.946	0.465	0.948	0.728
BigVul	0.626	1.000	1.000	0.816	0.861
MegaVul	0.794	0.993	0.411	0.884	0.771
Diversevul	1.000	0.786	0.930	0.900	0.904
D2A	0.635	0.328	0.050	0.642	0.414

REFERENCES

[1] P. Chakraborty, K.K. Arumugam, M. Alfadel, M. Nagappan, and S. McIntosh. Revisiting the performance of deep learning-based vulnerability detection on realistic datasets. 2024.

[2] S Chakraborty, R Krishna, Y Ding, and B Ray. Deep learning based vulnerability detection: Are we there yet? *IEEE Transactions on Software Engineering*, 48(9):3280–3296, 2022.

[3] Y Chen, Z Ding, L Alowain, X Chen, and D Wagner. Diversevul: A new vulnerable source code dataset for deep learning based vulnerability detection. 2023. Available: https://arxiv.org/abs/2304.00409.

[4] B Chernis and R Verma. Machine learning methods for software vulnerability detection. In R Verma and M Kantarcioglu, editors, *Proc. 4th ACM Int'l Workshop on Security and Privacy Analytics, IWSPA@CODASPY 2018, Tempe, AZ, USA, March 2018*, pages 31–39. ACM, 2018.

[5] J Fan, Y Li, S Wang, and T 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, New York, NY, USA, 2020. ACM.

[6] D Friedman and A Dieng. The vendi score: A diversity evaluation metric for machine learning, 2023.

[7] J Harer, L Kim, R Russell, et al. Automated software vulnerability detection with machine learning. *CoRR*, abs/1803.04497, 2018.

[8] Z Li, D Zou, S Xu, X Ou, H Jin, S Wang, Z Deng, and Y Zhong. Vuldeepecker: A deep learning-based system for vulnerability detection. *CoRR*, abs/1801.01681, 2018.

[9] C Ni, L Shen, X Yang, Y Zhu, and S Wang. Megavul: A c/c++ vulnerability dataset with comprehensive code representations. In *Proceedings of the 21st International Conference on Mining Software Repositories*, pages 738–742. ACM, 2024.

[10] C Northcutt, L Jiang, and I Chuang. Confident learning: Estimating uncertainty in dataset labels. *J. Artif. Int. Res.*, 70:1373–1411, May 2021.

[11] O Pieczul and S Foley. Runtime detection of zero-day vulnerability exploits in contemporary software systems. In *Data and Applications Security and Privacy XXX*, volume 9766 of *LNCS*, pages 347–363. Springer, 2016.

[12] F Yamaguchi, N Golde, D Arp, and K Rieck. Modeling and discovering vulnerabilities with code property graphs. In *IEEE symposium on security and privacy*, pages 590–604, 2014.

[13] Y Zheng, S Pujar, B Lewis, L Buratti, E Epstein, B Yang, J Laredo, A Morari, and Z Su. D2A: A dataset built for ai-based vulnerability detection methods using differential analysis. *CoRR*, abs/2102.07995, 2021.

[14] X Zhou and R Verma. Software vulnerability detection via multimodal deep learning. In Gabriele Lenzini and Weizhi Meng, editors, *Security and Trust Management*, pages 85–103. Springer International Publishing, Cham, 2023.

[15] Y Zhou, S Liu, J Siow, X Du, and Y Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. *CoRR*, abs/1909.03496, 2019. Available: http://arxiv.org/abs/1909.03496.

[16] X Zhu, S Wen, S Camtepe, and Y Xiang. Fuzzing: a survey for roadmap. *ACM Comp. Sur. (CSUR)*, 54(11s):1–36, 2022.

Software Vulnerability Detection: A Grand Challenge for AI

Aayush Gupta, Arthur Dunbar, Rakesh Verma
Dept. of Computer Science, University of Houston

1. The Grand Challenge

- Software bugs can lead to catastrophic failures – from spacecraft crashes to cybersecurity breaches.
- Traditional detection methods like fuzzing or manual review are incomplete.
- Machine learning and even LLMs have yet to solve this effectively.

2. Datasets

- We analyzed 9 datasets: **BigVul, Devign, DiverseVul, D2A, MegaVul, MVDSC, ReVeal, RealVul, and VulDEEPecker (VDP).**
- Each dataset has unique strengths (e.g., size, CWE diversity, realism) and weaknesses (e.g., duplication, imbalance).
- Existing datasets are **noisy, inconsistent**, or lack generalization.
- As shown in Figure 3, these datasets originate from common sources like SARD, QEMU, and Juliet, indicating limited diversity.

3. Models & Approach

- **4 Classical ML**: Logistic Regression, SVM, Random Forest, AdaBoost.
- **3 DL models**: DistilBERT, CodeBERT, RCNN.
- All models were evaluated on **F1 Score** across datasets in Table 3.
- DL models benefit from semantic understanding.

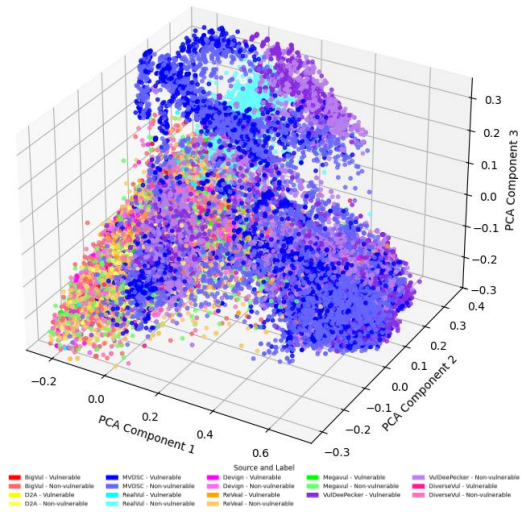

Figure 1. 3D PCA visualization of TF-IDF-encoded functions from the chosen 9 datasets. Color–coded by vulnerability label, the plot shows clustering patterns that indicate separability of vulnerability and non-vulnerability functions.

Dataset	Vulnerable	Non-Vulnerable
BigVul	14:10900	57:177736
Devign	1464:12460	648:14858
DiverseVul	3310:18945	7488:330492
D2A	1777:18653	158:1276970
MegaVul	36:17380	158:322168
MVDSC	5741:16142	17606:38858
ReVeal	479:2240	1924:20494
RealVul	2049:7555	78:88282
VDP	17723:17725	43888:43913

Table 1. Static analysis results from CPPCHECK across 9 SVD datasets. Each cell reports the number of code samples flagged with errors by CPPCHECK versus the total number of samples in that category *(flagged: total)*.

Dataset	CDS	CRS	VS	CCS	TS
ReVeal	0.960	0.992	0.511	0.729	0.798
Devign	0.671	0.905	0.371	1.000	0.737
MVDSC	0.249	0.661	0.033	0.457	0.350
VDP	0.260	0.452	0.035	0.530	0.319
RealVul	0.552	0.946	0.465	0.948	0.728
BigVul	0.626	1.000	1.000	0.816	0.861
MegaVul	0.794	0.993	0.411	0.884	0.771
Diversevul	1.000	0.786	0.930	0.900	0.904
D2A	0.635	0.328	0.050	0.642	0.414

Table 2. Relative Scoring of Datasets. Each dataset is scored on four metrics. Classifier Difficulty Score (CDS): how difficult it is to classify using simple classifiers. CleanLab Results Score (CRS): based upon scores from CleanLab. Vendi Score (VS): How much diversity the dataset has. Code Construct Coverage Score (CCS): How many code constructs does this dataset represent? These four metrics are averaged to give the Total Score (TS).

4. Results & Insights

- Classical ML still fails on most datasets.
- **DistilBERT** and **CodeBERT** show promise.
- **DiverseVul** and **BigVul** achieve the highest total quality scores, combining diversity, coverage, and low noise.
- **D2A** and **VDP** exhibit high duplication, label noise, and outliers.
- **Total Score (TS)** was computed by averaging four key metrics: CDS (difficulty), CRS (noise), VS (diversity), and CCS (code coverage).
  - Table 2 summarizes these scores – **DiverseVul leads**, while MVDSC and D2A lag behind

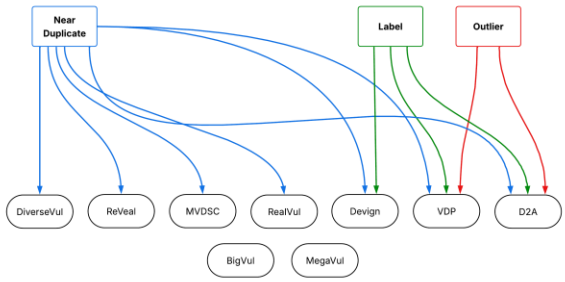

Figure 2. CleanLab based quality issues in the datasets. Only datasets with ≥ 5% incidence of Near Duplicate (blue), Label Inconsistencies (green), or Outliers (red) have been connected.

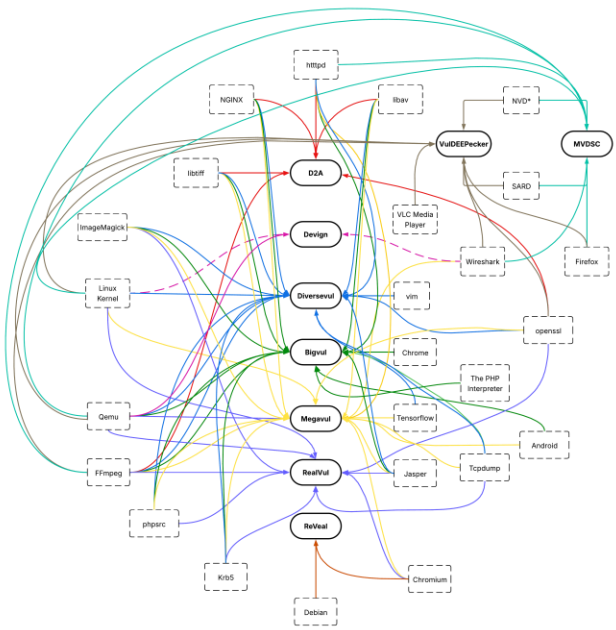

Figure 3. Relationship between foundational datasets and derived datasets. This visualization highlights how core datasets like SARD, Juliet, QEMU, and FFMpeg serve as the foundational sources (shown as dotted rectangular nodes) for widely used derived datasets like MVDSC, DiverseVul, and MegaVul (shown as rounded rectangular nodes). The colored directional arrows indicate the flow of contribution from each source to the resulting dataset.

Model / Dataset	ReVeal	Devign	MVDSC*	VDP*	RealVul Sampled*	BigVul	Megavul	Diversevul	D2A Sampled	Average
Logistic Regression	9.40	51.12	80.85	86.38	0.48	10.91	25.08	5.05	49.64	**35.43**
Adaboost	5.97	13.21	60.44	56.47	90.18	0.0	18.46	0.0	49.64	**32.71**
Random Forest	14.41	44.00	88.45	86.76	14.03	24.73	38.52	14.54	55.48	**42.32**
SVM	13.06	48.82	89.63	93.46	9.68	20.50	34.66	8.14	49.64	**40.84**
DistilBERT	41.09	63.32	95.00	95.46	98.79	92.55	45.00	63.32	25.37	**68.88**
CodeBERT	46.85	61.76	94.45	96.08	99.63	90.65	46.56	23.13	22.02	**64.57**
RCNN	9.10	26.60	45.74	33.58	65.00	95.65	28.28	2.41	7.70	**34.9**
Min DL - Max CL	-19.26	-24.49	-44.47	-55.19	-25.18	66.14	-10.22	-11.75	-47.78	**-19.13**
Max DL - Max CL	32.44	12.20	5.37	2.62	9.45	70.92	8.04	48.78	-30.11	**17.08**
Score	0.9601	0.6705	0.2493	0.2602	0.5523	0.6256	0.7944	1.0000	0.6351	**0.6386**
Average	**19.98**	**44.12**	**79.22**	**78.31**	**53.97**	**47.86**	**33.79**	**16.66**	**37.07**	

Table 3.F1 Scores for Classical and Deep Learning Models across datasets (* datasets are the cleaned version). The Classification Difficulty Score (CDS) is determined by subtracting the average F1 Score across all models from 1. This score is meant to capture how difficult the dataset is to classify using simple methods.