MADE-WIC: Multiple Annotated Datasets for Exploring Weaknesses In Code

Moritz Mock
Free University of Bozen-Bolzano
Bolzano, Italy
moritz.mock@student.unibz.it

Jorge Melegati
Free University of Bozen-Bolzano
Bolzano, Italy
jorge.melegati@unibz.it

Max Kretschmann
Hamburg University of Technology
Hamburg, Germany
max.kretschmann@tuhh.de

Nicolás E. Díaz Ferreyra
Hamburg University of Technology
Hamburg, Germany
nicolas.diaz-ferreyra@tuhh.de

Barbara Russo
Free University of Bozen-Bolzano
Bolzano, Italy
barbara.russo@unibz.it

ABSTRACT

In this paper, we present MADE-WIC, a large dataset of functions and their comments with multiple annotations for technical debt and code weaknesses leveraging different state-of-the-art approaches. It contains about 860K code functions and more than 2.7M related comments from 12 open-source projects. To the best of our knowledge, no such dataset is publicly available. MADE-WIC aims to provide researchers with a curated dataset on which to test and compare tools designed for the detection of code weaknesses and technical debt. As we have fused existing datasets, researchers have the possibility to evaluate the performance of their tools by also controlling the bias related to the annotation definition and dataset construction. The demonstration video can be retrieved at https://www.youtube.com/watch?v=GaQodPrcb6E.

CCS CONCEPTS

• **Software and its engineering** → *Maintaining software.*

KEYWORDS

Dataset annotation, SATD, security, vulnerabilities

ACM Reference Format:
Moritz Mock, Jorge Melegati, Max Kretschmann, Nicolás E. Díaz Ferreyra, and Barbara Russo. 2024. MADE-WIC: Multiple Annotated Datasets for Exploring Weaknesses In Code. In *39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24), October 27-November 1, 2024, Sacramento, CA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3691620.3695348

1 INTRODUCTION

Existing public datasets on software data typically provide their own schema and annotation method for a specific learning task (e.g., detection of code weaknesses). The schema is often determined by the goal of the research, and labels are typically generated manually, with heuristics derived for the available data or by statistic analyzers, which are not always accurate [10]. Differences in schema and annotation may prevent study replication and generalization. This is, for instance, the case of datasets annotated for vulnerability detection, for which literature reports several issues. Vulnerability datasets often rely on human-labelled techniques (e.g., commit differential analysis) that are resource-intensive [22]. A significant number (more than 60%) of vulnerabilities miss any annotation in practice, for example, due to silent fixes (i.e., developers commit changes to fix vulnerabilities but do not label/report the commits [16]), which implies that the actual number of vulnerabilities is much more than what it can be. Automated labelling results in a high percentage of incorrectly labelled vulnerabilities [16]. Finally, the different types of annotation techniques may produce different annotated datasets and detection results [5]. Another example are techniques to annotate self-admitted technical debt (SATD), i.e., annotate comments and related code (i.e., technical debt) that has low quality and requires future effort for refactoring [21]. Fig. 1

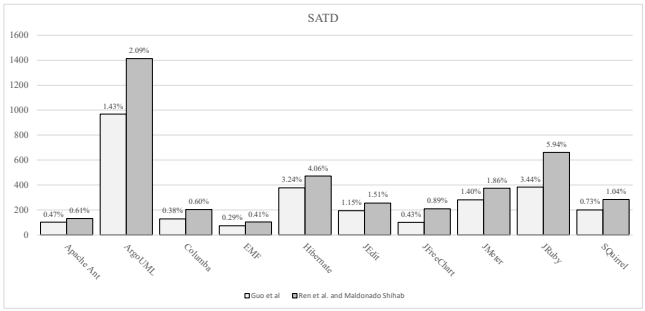

Figure 1: Differences in number and percentage of SATD instances in Ren *et al.* [20] and Guo *et al.* [9] on the same set of comments [12]

illustrates the difference in size of the same datasets annotated with the same technique by two different authors Guo *et al.* [9] and Ren *et al.* [20]. The figure reports the number of SATD instances (y-axis) and the percentage of SATD in the respective project. For instance, there is a difference of 2.5% in SATD instances for JRuby project. Differences in datasets' construction and annotation can produce different detection performances, as shown in Fig. 2. In this paper, we apply data fusion [1] to three existing datasets (WeakSATD [21],

ASE '24, October 27-November 1, 2024, Sacramento, CA, USA

ACM ISBN 979-8-4007-1248-7/24/10.
https://doi.org/10.1145/3691620.3695348

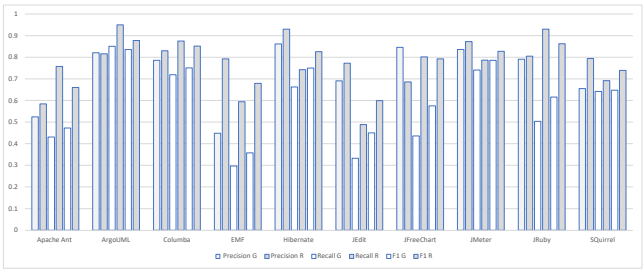

Figure 2: Differences in performance reported by Guo *et al.* of the Ren *et al.* approach [9] and in the original work of Ren *et al.* [20]

Devign [23], and Big-Vul [6]) and build MADE-WIC thereafter, a novel curated dataset of functions and comments (including leading comments, i.e., those comments preceding and related to a function), labelled for vulnerability, technical debt and security concerns. The dataset provides a unique schema and different annotations for the same instances and the above attributes. MADE-WIC contains about 860k functions and 2.7M of comments from 12 projects. We also propose an approach for the construction and annotation compliant with the schema of MADE-WIC that enables extension to further projects. The dataset, including the code to create it, is publicly available in the replication package [15].

2 ANNOTATION APPROACHES

We implemented different techniques with which we labelled the functions of MADE-WIC either directly or through related comments as described below.

2.1 Function Annotations

vf: We indicate with vf the original annotations of Devign and Big-Vul as described in the following.

Devign: The Devign approach [23] leverages a list of security-related keywords in commits' messages to classify commits as vulnerable, collects the previous version of the functions changed in the commits and then manually annotates them for vulnerability.

Big-Vul: The annotation method used in Big-Vul [6] parses the information in the Common Vulnerability Enumeration (CVE) repository [14] to retrieve the CVE entries that have reference links to publicly available Git repositories (e.g., GitHub) and their related bugs. From the related BugID, the fix commits and the previous versions of changed functions are retrieved and labelled vulnerable.

W: The W annotation uses heuristics extracted from the MITRE Common Weakness Enumeration repository (CWE) [13], as proposed in WeakSATD approach [21]. These are a set of rules that implement the descriptions provided in the CWE reports. We used the rules available in the replication package of WeakSATD [21].

2.2 Function Annotation through Comments

PS: The PS annotation labels functions as technical debt if they have at least one comment annotated with one of the 64 SATD patterns [19]. The patterns have been identified by manually inspecting more than 100,000 Java comments.

MAT: The MAT annotation leverages the Matches task Annotation Tags (MAT) heuristics [9] to label functions as technical debt if they have at least one comment annotated with one MAT tag *TODO, FIXME, XXX, and HACK*. MAT tags have been identified by exploring the default syntax highlighting of different IDEs.

SecI: The SecI annotation first automatically label functions as vulnerable if they have at least one comment annotated with one of the 288 security indicators [3]. Then, functions and their annotations have been manually reviewed by three of the authors, achieving a Fleiss' Kappa [8] score of 0.735 for their agreement. The result is a set of 89 agreed security indicators.

3 DATASET FUSION

MADE-WIC fuses and extends two existing datasets Devign [23], and Big-Vul [6] by creating a unique schema that transforms the data present in the sources into a common representation. The selected datasets and projects use C/C++, provide functions and comments or references to the GitHub projects from which to extract them, and make their annotation approach publicly available. The resulting dataset consists of functions and their comments (internal and leading comments). Fig. 3 illustrates the overall fusion and annotation approach as described in the following sections.

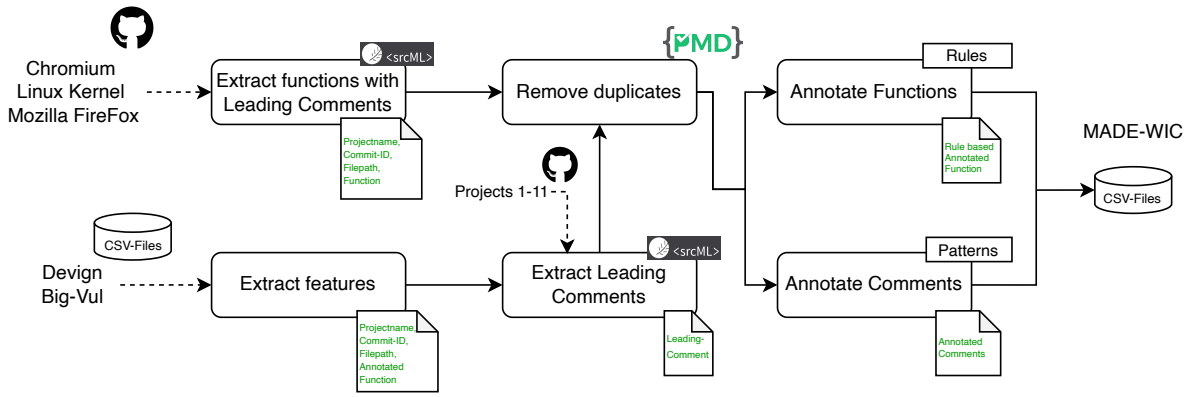

Figure 3: Fusion approach to generate MADE-WIC, extracting the information from existing datasets and open source projects.

Table 1: MADE-WIC schema.

	Name	Type	Description
Data	Project-name	string	Project from which the function was extracted
	Commit-ID	string	Commit hash from which the function was extracted
	Filepath	string	Path of the file that contains the function in the selected commit
	Function	string	Function code
	Leading-Comment	string	Comment preceding and related to the function
Annotation	PS	boolean	Flag indicating PS annotation
	MAT	boolean	Flag indicating MAT annotation
	Big-Vul	boolean	Flag indicating Big-Vul annotation
	Devign	boolean	Flag indicating Devign annotation
	W	boolean	Flag indicating W annotation
	SecI	boolean	Flag indicating SecI annotation

3.1 Data Extraction:

To extract the data, we implemented and automated two processes: 1) from the annotated datasets Devign and BigVul and 2) from the open-source repositories starting from the Chromium project used in WeakSATD and extended it to Linux Kernel, Mozilla FireFox.

From annotated datasets. From Devign and BigVul, we extracted the features (Project Name, Commit-ID, File Name, Annotation, Function) and used the commit-ID to get the project version. We then used srcML [2] to obtain any leading comment of the functions. Given that retrieving the leading comments is time-consuming, and Big-Vul is a very large dataset, we choose the ten projects with the largest number of vulnerabilities, accounting for 75% of the total. The projects are listed in Table 2. From the table, we can also see that three of the projects (Linux, Chrome, and FFmpeg) are shared between Devign and BigVul, but the different extraction and annotation techniques make them different in size and composition.

From open source public repositories. From open source projects (Chromium, Linux Kernel, Mozilla FireFox), we cloned the repositories at the last commit [1] and extract with srcML the features (Filepath, Annotation, Function, and leading comment, if any).

Finally, in both processes, we removed duplicates using PMD-CPD [18], which tokenizes the functions and calculates their similarity based on a threshold of 30 tokens. The threshold balances computational effort with the thoroughness of the inspection and includes 85% of all functions. Then, we annotated the functions in multiple ways, either directly or through the comments as described in Section 2.1. The annotated functions and their comments are then stored in the replication package in three CSV files called OSPR, BigVul, and Devign.

3.2 Schema

Table 1 illustrates the schema resulting from the fusion: each row describes a column of a CSV file of MADE-WIC. The schema contains eleven attributes. The annotations' flag indicates the type of annotation as described in Section 2.1. The annotations Big-Vul and Devign are extracted from the original datasets. The attributes *Project-name*, *Commit-ID*, and *Filepath* can be used to verify or extend in the future MADE-WIC. Table 2 describes the three CSV files compounding MADE-WIC. For each file, we report: (fn) the number of functions, (vf) the number of vulnerable functions as in the original dataset, (W) the number of vulnerable functions annotated with W, and the number of functions annotated as technical debt (with PS and MAT) and as security concerns (SecI). In particular, the annotations W, PS, MAT, and SecI are performed on all data in the dataset, see Table 2. Thus, the same schema and annotations allow for a comparison of existing datasets like Devign and BigVul that controls bias related to the construction of the annotated datasets when they are used for vulnerability, technical debt, or security detection.

[1] commit hash 57f97b2 for Chromium, e2ca6ba for Linux Kernel, and 4d46db3ff28b for Mozilla Firefox.

3.3 Application Scenarios

Our dataset can be used for various classification tasks that involve functions and/or their comments. For instance, it can be used to fine-tune pre-trained deep learning models for text and code (e.g., BERT-based transformers such as CodeBERT [7]) for downstream tasks that classify functions as technical debt and/or vulnerable by simply exploiting MADE-WIC with the PS annotation for technical debt and W for weakness. Studies can also understand the impact of different annotation techniques on the same data by, for instance, comparing MADE-WIC on vf and W, or the different subsets of MADE-WIC, Devign and BigVul, with their original vf annotations. MADE-WIC can also be used for the summarization of specific comments (e.g., generating SATD or security-related comments from functions) and masking tasks on functions and/or comments. For instance, researchers can mask PS patterns in comments or W patterns in functions and compare the ability of different transformers to retrieve back the patterns.

3.4 Data Quality

In this section, we leverage the work of Croft *et al.* [4] to assess the quality of MADE-WIC. The paper provides a set of attributes for high-quality of software vulnerability datasets.

Accuracy - The degree to which the data has attributes that correctly represent the true value of the intended attribute of a concept or event. To ensure accuracy, we first decided to use the state-of-the-art annotations. Then, we manually reviewed 31 functions annotated as vulnerable by W (one function per heuristic of the W annotation) and their comments from the OSPR portion. Each of the 31 functions was independently verified by three authors. In the end, 6 heuristics have been tuned and 10 removed. The authors then agreed on 21 rules and corresponding vulnerable functions.

Uniqueness - The degree to which there is no duplication in records. To remove duplicated functions, we used PMD-CPD [18] and a 99% threshold for the Jaccard index of overlapping function code. This process was carried out for each of the datasets, both for individual projects and between different projects. In the end, we removed 127 duplicates.

Consistency - The degree to which data instances have attributes that are free from contradiction and are coherent with other instances. As there is no oracle for the attributes we considered in this work,

Table 2: Functions (including their comments) in MADE-WIC: total (fn), vulnerability annotated as in original datasets (vf), vulnerability annotated as weaknesses (W), technical debt annotated as (PS) and (MAT), and security annotated as (SecI).

Project	fn	vf	W	PS	MAT	SecI
OSPR, language: C						
Chromium	20,028	-	5,205	151	508	514
Linux Ker.	652,726	-	210,220	5,594	8,444	13,182
Mozilla Fir.	15,380	-	4,200	116	436	141
Total	688,134	-	219,625	5,861	9,388	13,837
Devign, language: C						
FFmpeg	9,738	4,961	7,321	1,032	1,366	732
Qemu	17,544	7,476	7,785	477	1,378	653
Total	27,282	12,437	15,106	1,509	2,744	1,385
Big-Vul, language: C, C++						
Android	8,671	1,267	3,598	53	184	316
Chromium	77,167	3,938	10,434	165	569	334
FFmpeg	1,925	114	1,201	70	99	79
File(1) comm.	294	49	207	12	7	0
ImageMagick	2,489	338	1,271	12	11	91
Kerberos5	832	140	478	15	31	187
Linux Ker.	46,828	1,955	18,298	768	886	2,652
PHP Interp.	2,669	364	1,580	59	88	173
Radare2	1,168	73	722	19	67	19
Tcpdump	778	210	532	64	110	77
Total	144,358	8,448	38,214	1,237	2,042	3,928
Grand Total	**859,774**	**20,885**	**272,945**	**8,607**	**14,174**	**19,150**

the different annotation techniques for one attribute (e.g., vulnerability) have reported a portion of functions with positive and negative annotation depending on the technique. For instance, the percentage of functions that are consistently annotated vf and W ranges between 8% and 45%, while for technical debt (PS and MAT annotation), it ranges between 0% and 68% over individual projects.

Completeness - The degree to which subject data associated with an entity has values for all expected attributes and related instances. We provided multiple annotations for all attributes of interest (vulnerability, technical debt, security concerns). Some of the OSPR projects miss the Big-Vul or Devign original annotations and will be matter of future work.

Currentness - The degree to which data has attributes that are of the right age. The OSPR data was extracted and annotated in 2023 with the exception of the Chromium project, whose functions were extracted from the WeakSATD repository [21]. The Devign and Big-Vul datasets have been extracted from the original sources.

4 RELATED WORK AND CONCLUSIONS

To the best of our knowledge, MADE-WIC is the first dataset that provides functions, all relevant comments and their annotations for technical debt, vulnerability and security concerns. The three datasets we fused in this study (OSPR, Devign, Big-Vul) have single annotations or different annotation methods. For instance, WeakSATD [21] annotates for SATD files of the Chromium project with PS. In our work, we have reviewed this approach and extracted Chromium functions and comments from the original source. Not all the datasets in the literature could also be fused with our approach. For instance, the dataset of Lin *et al.* [11] that can be found in the replication package by Nong *et al.* [17] does not provide the commit-IDs from which the functions were extracted, preventing researchers from retrieving any further data than the ones they published, e.g., leading comment. D2A [22] is a dataset that leverages six open-source projects at function-level granularity. The annotation was done using a tool-based approach, focusing on the function versions before and after a fixing commit. Due to its large size of 3.7 GB, which includes more than 1.2 million instances, integrating this extensive dataset will be addressed in future work.

Acknowledgments: Moritz Mock is partially funded by the National Recovery and Resilience Plan (Piano Nazionale di Ripresa e Resilienza, PNRR - DM 117/2023). The work has been funded by project no. EFRE1039 under the 2023 EFRE/FESR program. We acknowledge the CINECA award under the ISCRA initiative, for the availability of high-performance computing resources and support.

REFERENCES

[1] Jens Bleiholder and Felix Naumann. 2009. Data fusion. *ACM Comput. Surv.* 41, 1, Article 1 (jan 2009), 41 pages.
[2] Michael L. Collard, Michael John Decker, and Jonathan I. Maletic. 2013. srcML: An Infrastructure for the Exploration, Analysis, and Manipulation of Source Code: A Tool Demonstration *(ICSME)*. 516–519.
[3] Roland Croft et al. 2022. An empirical study of developers' discussions about security challenges of different programming languages. *Empirical Software Engineering* 27 (2022), 1–52.
[4] Roland Croft, M. Ali Babar, and M. Mehdi Kholoosi. 2023. Data Quality for Software Vulnerability Datasets *(ICSE)*. 121–133.
[5] Roland Croft, Yongzheng Xie, and Muhammad Ali Babar. 2022. Data preparation for software vulnerability prediction: A systematic literature review. *IEEE Transactions on Software Engineering* 49, 3 (2022), 1044–1063.
[6] Jiahao Fan, Yi Li, Shaohua Wang, and Tien N. Nguyen. 2020. A C/C++ Code Vulnerability Dataset with Code Changes and CVE Summaries *(MSR '20)*. Association for Computing Machinery, New York, NY, USA, 508–512.
[7] Zhangyin Feng et al. 2020. CodeBERT: A pre-trained model for programming and natural languages. *EMNLP 2020* (2020), 1536–1547.
[8] Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin* 76, 5 (1971), 378.
[9] Zhaoqiang Guo et al. 2021. How Far Have We Progressed in Identifying Self-Admitted Technical Debts? A Comprehensive Empirical Study.
[10] Kim Herzig and Andreas Zeller. 2013. The impact of tangled code changes *(MSR)*. IEEE, 121–130.
[11] Guanjun Lin, Wei Xiao, Jun Zhang, and Yang Xiang. 2020. Deep Learning-Based Vulnerable Function Detection: A Benchmark. In *Information and Communications Security*, Jianying Zhou, Xiapu Luo, Qingni Shen, and Zhen Xu (Eds.). Springer International Publishing, Cham, 219–232.
[12] Everton da Silva Maldonado, Emad Shihab, and Nikolaos Tsantalis. 2017. Using Natural Language Processing to Automatically Detect Self-Admitted Technical Debt. *IEEE Transactions on Software Engineering* 43, 11 (2017), 1044–1062.
[13] MIRTE. 2006. Common Weakness Enumaration. https://cwe.mitre.org
[14] MITRE. 1999. Common Vulnerabilities and Exposures. https://cve.mitre.org
[15] Moritz Mock, Jorge Melegati, Max Kretschmann, Nicolás E. Díaz Ferreyra, and Barbara Russo. 2024. *Replication package.* https://doi.org/10.5281/zenodo.13370805
[16] Giang Nguyen-Truong et al. 2022. HERMES: Using Commit-Issue Linking to Detect Vulnerability-Fixing Commits *(SANER)*. 51–62.
[17] Yu Nong et al. 2023. Open Science in Software Engineering: A Study on Deep Learning-Based Vulnerability Detection. *IEEE Transactions on Software Engineering* 49, 4 (2023), 1983–2005.
[18] PMD 2024. *PMD-CPD.* https://pmd.github.io/pmd/pmd_userdocs_cpd.html
[19] Aniket Potdar and Emad Shihab. 2014. An Exploratory Study on Self-Admitted Technical Debt *(ICSME)*. 91–100.
[20] Xiaoxue Ren et al. 2019. Neural Network-Based Detection of Self-Admitted Technical Debt: From Performance to Explainability. *ACM Trans. Softw. Eng. Methodol.* 28, 3, Article 15 (jul 2019), 45 pages.
[21] Barbara Russo, Matteo Camilli, and Moritz Mock. 2022. WeakSATD: Detecting Weak Self-admitted Technical Debt *(MSR)*. 448–453.
[22] Yunhui Zheng et al. 2021. D2a: A dataset built for ai-based vulnerability detection methods using differential analysis *(ICSE-SEIP)*. IEEE, 111–120.
[23] Yaqin Zhou et al. 2019. *Devign: Effective Vulnerability Identification by Learning Comprehensive Program Semantics via Graph Neural Networks.* Curran Associates.