New Version, New Answer: Investigating Cybersecurity Static-Analysis Tool Findings

1st A.M. Reinhold
ORCID 0000-0003-0411-3486

2nd Travis Weber
ORCID 0000-0003-0444-0997

3rd Colleen Lemak
ORCID 0000-0001-8064-0806

4th Derek Reimanis
ORCID 0000-0002-0747-0457

5th Clemente Izurieta
ORCID 0000-0002-1002-3906

***Abstract*—Automated detection of vulnerabilities and weaknesses in binary code is a critical need at the frontier of cybersecurity research. Cybersecurity static-analysis tools aim to detect and enumerate vulnerabilities and weaknesses. Two popular tools are CVE Binary Tool (cve-bin-tool) and cwe-checker. Cve-bin-tool reports vulnerabilities using Common Vulnerabilities and Exposures (CVE) whereas cwe-checker reports weaknesses using Common Weakness Enumeration (CWE). Despite widespread use, the consistency with which these tools report vulnerabilities and weaknesses (herein, "findings") was unaddressed. We conducted a systematic investigation of 660 unique binaries taken from a Kali Linux distribution, evaluated each binary with multiple versions of the static-analysis tools, and investigated how the findings changed according to the version of the static-analysis tool used. We expected some variation in findings commensurate with the software-development life cycle. However, we were surprised by the number and magnitude of the changes in findings reported across versions. New versions gave new answers.**

***Index Terms*—binary, code security, detection, CWE, CVE**

I. Introduction

Static-analysis tools report warnings and statistics about a program without executing the program. In the domain of cybersecurity, static-analysis tools detect potential security threats in a program, including weaknesses and vulnerabilities. CVE Binary Tool ("cve-bin-tool" developed by Intel; [1]) and cwe-checker (developed by German research organization Fraunhofer FKIE; [2]) are two commonly-employed static-analysis tools that provide an objective evaluation of the vulnerabilities and weaknesses in program binaries, respectively.

A.M. Reinhold, Derek Reimanis, and Clemente Izurieta: Funded by the Department of Homeland Security (DHS) Science and Technology (S&T) Directorate under contract number 70RSAT22CB0000005. Travis Weber and Colleen Lemack: Portions of this material are based upon work supported by the United States National Science Foundation under Grant No. CCF-1947750. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation and DHS S&T.

A.M. Reinhold, Derek Reimanis, and Clemente Izurieta are with the Software Engineering and Cybersecurity Laboratory, Gianforte School of Computing, Montana State University, Bozeman, MT 59717 USA (email: reinhold@montana.edu). Travis Weber and Colleen Lemack were 2022 NSF REU Summer Scholars in the Gianforte School of Computing, Montana State University, Bozeman, MT 59717 USA.

Cwe-checker and cve-bin-tool have the common purpose of assessing program security but operate differently. Cwe-checker decompiles binary files using the United States' National Security Agency Reverse Engineering Framework, ghidra [3], for the purpose of identifying buggy or weakness-prone classes in the program. Cwe-checker uses the Common Weakness Enumeration catalog [4] to classify potential weaknesses, reported as CWEs [5].

In contrast, cve-bin-tool [1] checks for strings via pattern matching within a binary file. Specifically, cve-bin-tool performs regular expression matching between the bytes in a binary file and signatures of many versions of commonly-used compiled programs, such as mysql [6] and nano [7]. In the event of a positive regular expression match, cve-bin-tool determines that the compiled program exists within the binary under analysis, and then queries the National Vulnerabilities Database (NVD) [8] and the Open Source Vulnerabilities Database (OSV) [9] to collect all known vulnerabilities (CVEs; Table I) associated with a particular compiled program (e.g., mysql or nano).

Within binary analysis, cwe-checker and cve-bin-tool operate at the frontier of cybersecurity research and offer a promising solution to the challenging problem of identifying and enumerating an incredibly vast suite of potential threats. Decision-making based on output from these tools presupposes that they provide a direct measure of key aspects of code security. However, the readme of both tools features a disclaimer stating that the results of the tools should be used at an end-user's discretion. The developers of these tools do not claim that the tools work perfectly.

As users of these binary static-analysis tools who make decisions based on their outputs, we sought to understand the consistency and reliability with which cwe-checker and cve-bin-tool report findings from one version to the next. We began by combing the academic literature; however, after surveying the IEEE Archives [10], Google Scholar [11], and Web of Science [12], we found no studies that systematically evaluated the consistency or reliability in which these tools report "findings" (defined in Table I). Thus, our research addresses the question: *How consistent are the outputs of cwe-checker and cve-bin-tool across versions of each tool?* We predicted that some variation would be present between versions of the

TABLE I
DEFINITIONS

Term	Definition	Source(s)
vulnerability	program code that an attacker may exploit to gain access into a system or network	NVD [8]
weakness	code having the potential to develop into a software vulnerability	CWE catalog [4]
findings	herein, CVE occurrences reported by cve-bin-tool; CWE occurrences reported by cwe-checker	the authors

tools as, e.g., developers make updates to ensure relevance, but that large fluctuations in reported findings would not occur between sequential versions.

II. METHODS

To address our research question, we ran 660 publicly available binaries (herein, "collection of binaries") from a Kali Linux distribution through as many functioning versions of each tool as were available (collection of binaries can be downloaded from our github page at https://github.com/MSUSEL/tool-evolution and are described in detail in [13]). For cwe-checker, we acquired and reported results for three working versions: versions 0.4, 0.5, 0.6. Versions 0.1, 0.2, and 0.3 of cwe-checker required the installation of outdated and deprecated dependencies, which required substantive changes to environment configuration; consequently, our analysis herein focused on versions 0.4, 0.5, and 0.6. For cve-bin-tool, we acquired eleven working versions. Two working versions of cve-bin-tool were omitted because they appeared to have bugs; these versions reported identical scores for all 660 binaries. Thus, nine versions of cve-bin-tool were included in our results: versions 1.0, 1.1, 2.0, 2.1, 2.1post1, 2.2, 2.2.1, 3.0, 3.1.1.

Importantly, in order to be able to attribute the variation in tool output to differences in the versions of the static-analysis tools, we controlled for multiple factors. One, we assessed the *same version of each binary in the collection* with multiple versions of the static-analysis tools; thus, all versions of the static-analysis tools evaluated the exact same binary code. Two, to ensure that the differences in cve-bin-tool output were due to the differences in cve-bin-tool versions and not differences in the NVD or OSV, we used the NVD and OSV acquired on 18 July 2022. Thus, differences in tool output from different versions of cve-bin-tool are not the result of different versions of the NVD and OSV.

Data were analyzed using the R Language and Environment for Statistical Computing [14], using the "ggplot2" package [15] to generate all plots and selecting colors with the "viridis" package [16]. We visualized the variability in tool outputs across versions as follows. One, we plotted the total number of findings reported by each version of each tool for each binary (Fig. 1A-B). Two, we used Jenks natural breaks optimization, a 1-D clustering approach, to group the total number of findings (function "getJenksBreaks" in the "BAMMtools" package [17]). We explored a range of groups from $k = 3$ to $k = 12$. We selected $k = 8$ groups based on visual inspection; $k < 8$ groups resulted in compression of practical variability of the data whereas $k > 8$ resulted in visualizing variability in the data that lacked practical significance. We plotted the scores for each tool for each binary in an alluvial plot (Fig. 1C-D). For this and the subsequent effort, we used all working versions of cwe-checker, but focused on the major (i.e., ".0") releases of cve-bin-tool. Three, we calculated the differences in the number of findings reported by the static-analysis tools between versions and plotted these in symmetrical density (violin) plots (Fig. 1E-F).

We tested for the effect of version on the number of findings reported for each static-analysis tool using a Friedman test (function "friedman.test" in the "stats" package [14]). We used a Friedman test because the findings reported for each tool failed to meet the assumptions of parametric statistics, but needed to be modeled with a statistic that accounts for repeated measures. We ran one Friedman test for cwe-checker and another for cve-bin-tool. In both tests, tool findings were modeled as the dependent variable, versions of the static-analysis tools were modeled as the independent variable, and binaries were modeled as the blocking variable to account for the fact that each binary was evaluated with multiple versions of the static-analysis tool.

To understand the patterns underpinning the variation across versions of the static-analysis tools, we explored summary statistics in the tool outputs for each CWE (cwe-checker) and for each CVE prefix—which corresponds to the year that the CVE was cataloged (cve-bin-tool). Findings reported as "UNKN" indicate those wherein cve-bin-tool could not parse the output received from the NVD; cve-bin-tool reports these findings as "UNKNOWN", and they are included here (e.g., Figs. 2B and 3B). The plots of the summary statistics (standard deviations [Figs. 2 and 4] and medians [Fig. 3]) explore the CWEs and CVE-year prefixes driving the variation in the findings summarized in Fig. 1.

III. RESULTS

The outputs from the cybersecurity static-analysis tools varied according to the version of the tool (cwe-checker: Friedman chi-squared = 591.7, degrees of freedom [DF] = 2, P-value $< 2.2\text{e-}16$; cve-bin-tool: Friedman chi-squared = 4,715.5, DF = 8, P-value $< 2.2\text{e-}16$; Fig. 1). Although only one version of each binary was evaluated, the number of findings was rarely constant across all versions of the static-analysis tools.

Only 14.1% of of the 660 binaries evaluated were found to have the same number of findings by all versions of cwe-checker. The majority of these—90 of 93 binaries—were reported to have zero findings across all versions of cwe-checker. Results for cve-bin-tool were similar. Of the 660 binaries evaluated, only 2.3% of binaries were found to have the same number of findings by all versions of cve-bin-tool. The majority of these—14 of 15 binaries—were reported to have zero findings across all versions of cve-bin-tool. Thus,

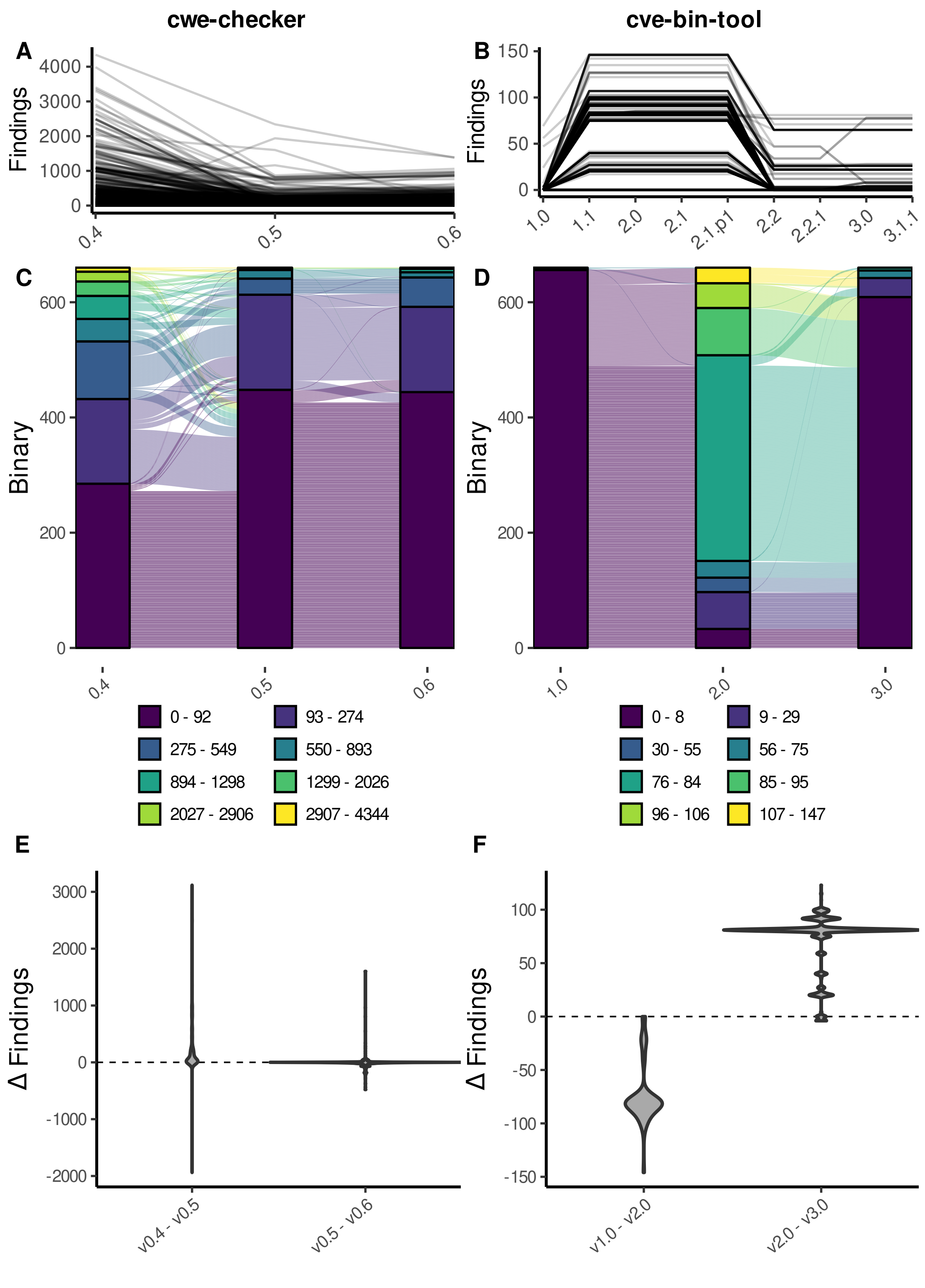

Fig. 1. Findings reported by different versions of cwe-checker and cve-bin-tool outputs for the collection of binaries. One version of each binary was run through multiple versions of each static-analysis tool; thus, variation in tool outputs is attributable to the differences in the versions of the static-analysis tools. **A-B**: Sum of reported findings versus version of static-analysis tool. Each line represents one of the binaries analyzed. **C-D**: Alluvial plots depicting changes in the sum of reported findings for the collection of binaries. The color of the stacked bars denotes the sum of findings reported by the version of the static-analysis tool indicated on the x-axis. Between stacked bars, each binary is represented by a thin line; lines connecting bars of different colors indicate binaries which were assessed as having different numbers of findings by different versions of the static-analysis tools. The ranges in findings associated with the color ramp were determined using Jenks natural breaks optimization (see *Methods*). **E-F**: Symmetrical density plots depicting mathematical differences in the number of findings associated with different versions of the static-analysis tools. For each binary, the sum of the findings reported by the newer version of the static-analysis tool was subtracted from the sum of the findings reported by the older version of the static-analysis tool. Thus, the density plots represent the distribution of differences in reported findings between the versions of static-analysis tools indicated on the x-axis.

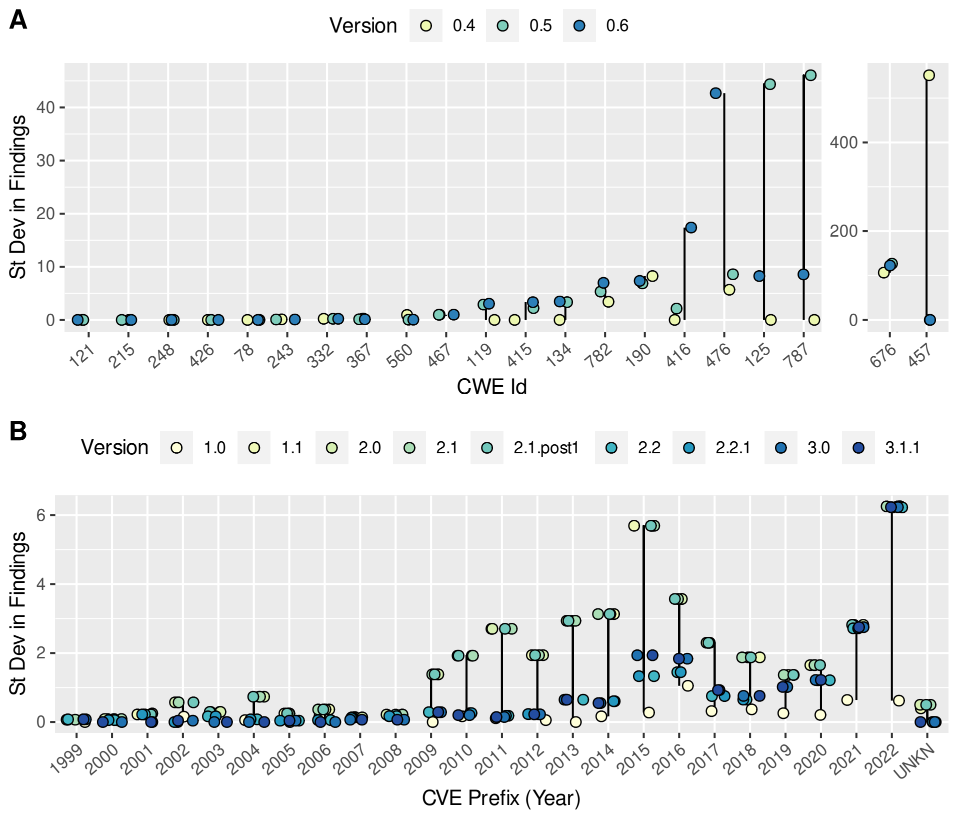

Fig. 2. Standard deviation in the number of findings identified by cwe-checker (**A**) and cve-bin-tool (**B**) versus CWE Id and CVE Prefix, respectively. Black vertical bars depict the range in standard deviation for the CWE or CVE prefix indicated on the x-axis; i.e., bar length indicates range of variability in static-analysis tool output. In panel A, the CWEs on the x-axis are sorted by the maximum standard deviation calculated for the collection of binaries across all versions of cve-bin-tool. In panel B, the CVE prefixes are sorted by year.

only three binaries were found to have consistent, nonzero scores across all versions of cwe-checker, and only one binary had consistent, nonzero scores across all versions of cve-bin-tool.

We explored how the reported number of findings changed from one version of the static-analysis tools to the next. The number of findings reported by cwe-checker was profoundly different for many binaries, depending on the version analyzing the binary code (Figs. 1A, C, and E). The median number of findings reported by cwe-checker was 66 higher in v0.4 than v0.5 and the same in both v0.5 and v0.6 (Fig. 1E). However, these medians belie the extent of the variability in the findings (Fig. 1C and E). The differences in the findings between v0.4 and v0.5 ranged from 1,939 fewer findings to 3,118 greater findings (standard deviation [SD] = 530.9); the differences in the findings between v0.5 and v0.6 ranged from 480 fewer findings to 1,601 greater findings (SD = 99.6).

With respect to the major releases, cve-bin-tool reported the majority of the binaries as having a higher number of findings in v2.0 than in either v1.0 or v3.0 (Fig. 1B, D, and F). More specifically, the median number of findings reported by cve-bin-tool was 81 fewer in v1.0 than in v2.0 but 81 higher in v2.0 than v3.0 (Fig. 1B and F). The differences in the findings between v1.0 and v2.0 ranged from 0 to 146 findings fewer (SD = 28.6); the differences in the findings between 2.0 and v3.0 ranged from 4 fewer findings to 123 greater findings (SD = 27.6).

A. *Detailed cwe-checker results*

Four CWEs underpinned the majority of the variation in the cwe-checker findings presented in Fig. 1A, C, and E. These were CWE-457, CWE-787, CWE-125, and CWE-476 (Figs. 2A and 3A). The substantive decrease in the total number of findings between v0.4 and v0.5 was primarily driven by CWE-457, which is found in 85% of binaries by v0.4 but no subsequent versions (Fig. 3A; see *Discussion* and Table II). In addition, v0.4 reported more variation in findings for CWE-457 than any other CWE in any version of cwe-checker (Fig. 2A). Cwe-checker v0.5 reported the greatest variation in findings for CWE-787 (followed closely by CWE-125); v0.6 reported the greatest variation in findings for CWE-476.

Cwe-checker v0.4 never found an instance of CWE-787 or CWE-125 in any of the binaries, but v0.5 and v0.6 found both of these CWEs in many binaries (Fig. 3A). CWE-787 was found in 49% of binaries by v0.5 and in 40% of binaries by v0.6. When found in a binary, CWE-787 was generally detected multiple times; multiple detections occurred in 79% of binaries found to have this CWE by v0.5 and 78% by v0.6.

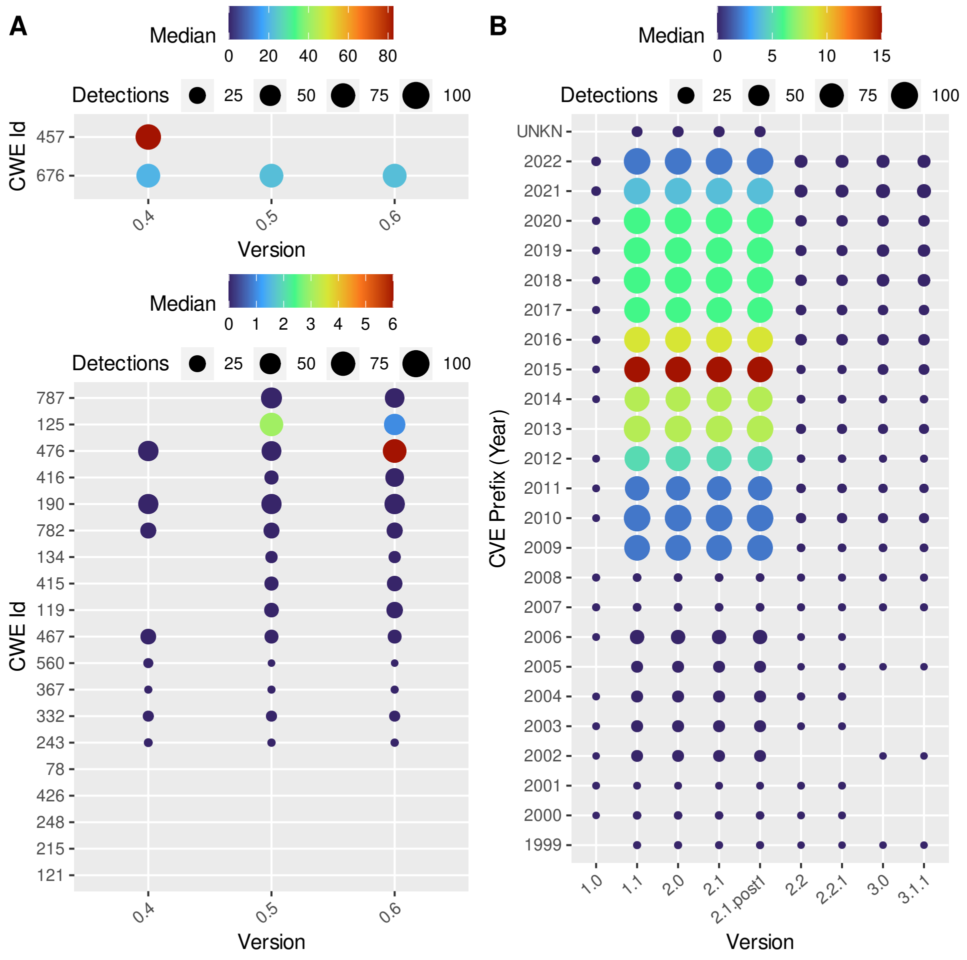

Fig. 3. Percent of files with detections of CWEs (**A**) and CVEs (**B**) versus version of cwe-checker and cve-bin-tool, respectively. Point size depicts the percent of binaries in which cwe-checker and cve-bin-tool detected the respective CWEs or CVEs indicated on the y-axis for each version of the static-analysis tool indicated on the x-axis; the absence of a point indicates that the CWE or CVE on the y-axis was not detected in the version of the static-analysis tool on the x-axis. Point color depicts the median number of findings within each binary for each CWE or CVE in each version of the static-analysis tool.

Although the median number of findings for CWE-787 was 0 for all versions of cwe-checker, associated SDs were 46.0 and 8.6 for v0.5 and v0.6, respectively (Fig. 2A). Thus, this CWE is not found consistently across versions of cwe-checker.

Cwe-checker v0.5 and v0.6 found CWE-125 in more binaries than CWE-787 (Fig. 3A). CWE-125 was found in 68% of binaries by v0.5 and 51% of binaries by v0.6. Similar to CWE-787, multiple detections of CWE-125 within the same binary was common; multiple detections occurred in 91% of binaries found to have this CWE by v0.5 and 79% by v0.6. Median findings for CWE-125 were 3 (SD = 44.4) and 1 (SD = 8.3) for v0.5 and v0.6, respectively (Fig. 2A). Thus, CWE-125 is not found consistently across versions of cwe-checker.

All versions of cwe-checker detected CWE-476 in the binaries, but detections varied according to the version of cwe-checker (Fig. 3A). Cwe-checker v0.4, v0.5, and v0.6 detected this CWE in 45%, 41%, and 70% of the binaries, respectively. Multiple detections of CWE-476 within the same binary was common; multiple detections occurred in 78% of binaries found to have this CWE by v0.4, 82% by v0.5, and 94% by v0.6. Median findings were 0 for v0.4 (SD = 5.7), 0 for v0.5 (SD = 8.6), and 6 for v0.6 (SD = 42.7; Fig. 2A). Thus, findings of CWE-476 are inconsistent across versions of cwe-checker.

B. *Detailed cve-bin-tool results*

Across versions, cve-bin-tool reported relatively low numbers of findings for CVEs dating from 1999-2008 (Fig. 3B). Commensurately, the variation in the number of findings was low for these older CVEs (SD $<$ 1; Fig. 2B). However, a starkly different pattern emerged for CVEs from years 2009-2022.

Across versions, cve-bin-tool reported highly variable findings for CVEs from years 2009-2022 (Figs. 2B and 3B). The ranges in the standard deviations of findings were larger for CVEs from 2009-2022 than from 1999-2008 (Fig. 2B); the versions of cve-bin-tool that contributed most to this result were v1.1 through v2.1post1. These versions of cve-bin-tool reported a greater number of binaries containing CVEs than other versions of cve-bin-tool (*Detections* in Fig. 3B). These versions of cve-bin-tool also had the highest median

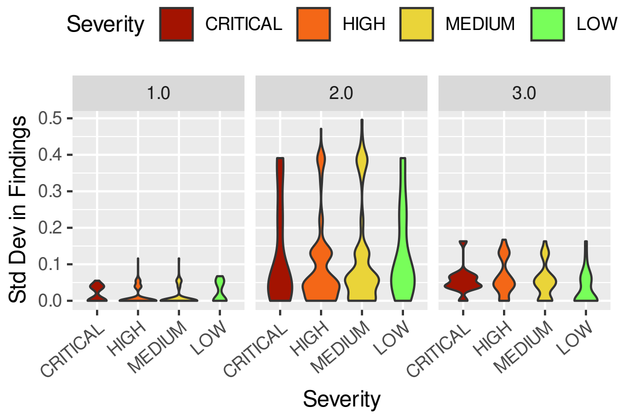

Fig. 4. Symmetrical density plots depicting the standard deviation in cve-bin-tool findings according to major-release version (panels) and CVE severity (colors).

number of findings for the CVEs from years 2009-2022, with a noticeable peak for CVEs cataloged in 2015. Thus, cve-bin-tool outputs were variable both in terms of reporting which binaries contained vulnerabilities and the number of vulnerabilities present in those binaries.

Because our team wanted to know if cve-bin-tool reported more consistent findings for CVEs of higher severity, we investigated the relationship between CVE severity and the standard deviation in reported findings (Fig. 4). We find no evidence in support of this claim. Rather, the version of cve-bin-tool appears a much stronger determinant of the reported findings than CVE severity.

IV. Discussion

Practitioners need confidence in the instrument fidelity of cybersecurity static-analysis tools. They need to know that the tools reliably identify and enumerate any vulnerabilities or weaknesses in code. Yet, this is a particularly challenging objective in binary analysis, with myriad potential sources of Type I and Type II errors. The observed variability in the tool outputs across versions underscores this challenge (Figs. 1, 2, 3). Unfortunately, this variability also hampers confidence in decisions made based on outputs from these tools.

With respect to cwe-checker, we had posited that differences in how the tool versions operate underpin differences in reported findings (Table II). Note that v0.4 aims to detect instances of CWEs 248 and 457, but no subsequent versions detect these two CWEs (see changelog from v0.4 to v0.5 [18]). From our detailed analyses, however, we know that only CWE 457 contributes to this pattern; CWE 248 was never detected (Fig. 3A).

Conversely, we had posited that increases in the number of findings between v0.4 and v0.5 were attributable to v0.5 evaluating code for five additional CWEs than v0.4 (Table II). Only three of the CWEs contributed to this pattern; these were CWEs 119, 415, and 416. CWEs 78 and 121 were never detected.

Likewise, we posited that differences in findings reported between v0.5 and v0.6 were attributable to differences in detection methods for CWEs 78, 119, and 416. The developers significantly changed how cwe-checker detects instances of these CWEs with the release of v0.6 [19]. However, we observed only small differences between v0.5 and v0.6 for CWEs 119 and 416; again, CWE 78 was never detected.

With respect to cve-bin-tool, we sought to understand the drivers of the rise in findings from v1.0 to v1.1 and the subsequent drop from v2.1post1 to v2.2. We extracted the version release notes from v1.1 and v2.2 in search of significant changes to the manner with which vulnerabilities are detected. We note that in v1.1 the developers wrote "This is a minor bug fix release to address an issue with the NVD download," suggesting that a bug in the NVD download process was affecting results prior to v1.1. Our results show that number of reported vulnerabilities increased dramatically in v1.1, which we attribute to the bug fix (Figs. 1, 2, 3).

While our research does not address the accuracy of cve-bin-tool directly, we interpreted that developer-stated bug fixes resulted in a more accurate tool. Following these bug fixes, cve-bin-tool results were generally consistent from v1.1 to v2.1post1. However, this changed with the release of v2.2 which was accompanied with the note "There are also a number of new checkers and bug fixes." Despite this increase in the number of checkers, the number of reported vulnerabilities decreased precipitously between v2.1post1 and v2.2. This decline in findings was surprising given the increase in the number of checkers. Thus, the decline in findings should be a result of bug fixes.

Our team had also posited that CVEs with higher severities would receive more attention by the developers. Supposing this was true, we predicted that cve-bin-tool results for CVEs with higher severities would have more consistent findings across the versions than CVEs with lower severities. However, our results did not support this contention (Fig. 4).

A. Implications for practitioners and end users

Our analyses suggest that cwe-checker and cve-bin-tool have not yet solved the numerous challenges associated with binary analysis. Neither static-analysis tool consistently finds weaknesses or vulnerabilities in code. For both tools, it remains unclear which findings are—and are not—actually present in the collection of binaries. While these tools are promising, they are not yet mature. More development, validation, and verification are required to improve the usefulness of cwe-checker and cve-bin-tool in a decision-making context.

Practitioners make decisions about software security based on newer versions of static-analysis tools, often assuming that latest releases are most relevant. However, our results indicate that this practice is precarious. Moving forward, our team will only make comparisons about software security across projects *if the same version of the same tool is used*. Likewise, we will only track the security of a binary over time *if the same version of the same tool is used*. We suggest practitioners do the same.

TABLE II
WEAKNESSES DETECTED BY THE VERSIONS OF CWE-CHECKER

Weakness	v0.4	v0.5	v0.6
CWE-78*	-	X	X
CWE-119*	-	X	X
CWE-121	-	X	X
CWE-125	X	X	X
CWE-134	X	X	X
CWE-190	X	X	X
CWE-215	X	X	X
CWE-243	X	X	X
CWE-248	X	-	-
CWE-332	X	X	X
CWE-367	X	X	X
CWE-415	-	X	X
CWE-416*	-	X	X
CWE-426	X	X	X
CWE-457	X	-	-
CWE-467	X	X	X
CWE-476	X	X	X
CWE-560	X	X	X
CWE-676	X	X	X
CWE-782	X	X	X
CWE-787	X	X	X

*Detection code altered from v0.5 to v0.6.

B. Threats to Validity

Because our analysis was highly controlled—we evaluated the same collection of binaries across all versions of the static-analysis tools and used the same NVD and OSV for all versions of cve-bin-tool—we attribute variation in tool output to differences in the versions of the static-analysis tools. We do not perceive significant threats to the repeatability of our investigation. Rather, the entire investigation can be repeated by following the data pipeline that will be permanently archived on Zenodo upon acceptance.

Great effort was made to ensure that the internal validity [20] of our investigation was sound; i.e., that the differences in tool outputs are the result of differences in the versions of the tools. However, if there are factors beyond the scope of what we controlled for, then a threat to internal validity is an issue. For instance, each version of the two static-analysis tools was run with the same version of Python (cwe-checker: v3.9.7; cve-bin-tool: v3.7.3). If the static-analysis tools provide different outputs on different versions of Python, then that is a threat to internal validity. Other potential threats to internal validity include whether the static-analysis tools report different numbers of findings depending on multithreading, operating system, or if changes in the data interpretation within the tool impact outputs (e.g., if the NVD or OSV updated the data structure that their API returns).

The external validity [20] of our investigation is limited to the collection of publicly available binaries that we acquired. Extrapolating our results beyond these binaries should be done with nuance and care. For instance, if another investigator should evaluate a binary in a "similar family" to the programs we evaluated, the potential threat to external validity is low. If, however, they evaluate a binary that is much larger, smaller, or of a totally different family than we assessed here, then we caution against that extrapolation.

C. Future Plans

Our results here raise two critical questions: *are the findings truly present in the binary code or not?* and *which versions of the tools are most accurate?* Our future research will address both of these questions, keeping an eye towards understanding the mechanisms that underpin the sources of variability in tool outputs and identifying the use cases where these tools can be used with confidence and where they cannot.

There were some CWEs and CVE prefixes for which detections and enumerations were consistent. For instance, CWE 676 was remarkably consistent across versions of cwe-checker. Discovering the reasons for this consistency was beyond the scope of our current work, but our future work will investigate reasons for this consistency and determine if consistency and accuracy are correlated. Further, we are planning a series of experiments wherein we will strategically inject code with specific weaknesses and vulnerabilities to assess where tool accuracy holds and where it falters. This research will be an asset to end users and developers alike. End users will know which vulnerabilities and weaknesses are consistently identified and developers can focus their attention on the vulnerabilities and weaknesses that are not identified with consistency.

D. Conclusions

The purpose of our work was to investigate the consistency with which cybersecurity static-analysis tools report weaknesses and vulnerabilities in binary files. This work fills a crucial gap as the consistency with which these tools report findings was—until now—not assessed in any systematic way. Our systematic analysis of several hundred binaries through multiple versions of well-used, well-known cybersecurity static-analysis tools is a first step to understanding the constraints on how these tools can be used with confidence. The variability in the reported findings of cve-bin-tool and cwe-checker causes us to use these tools with trepidation until we can better understand the sources of variation in their outputs.

REFERENCES

[1] Intel, "CVE Binary Tool (cve-bin-tool)," 2022. [Online]. Available: https://github.com/intel/cve-bin-tool

[2] Fraunhofer FKIE, FKIE-CAD, "cwe_checker," 2022. [Online]. Available: {https://github.com/fkie-cad/cwe{_}checker}

[3] National Security Agency, "Ghidra software reverse engineering framework." [Online]. Available: https://github.com/NationalSecurityAgency/ghidra/

[4] Mitre Corporation, "Common weakness enumeration: a community-developed list of software and hardware weakness types." [Online]. Available: https://cwe.mitre.org/

[5] S. Christey, J. Kenderdine, J. Mazella, and B. Miles, "The evolution of the CWE development and research views," *Mitre Corporation*, 2008. [Online]. Available: https://cwe.mitre.org/documents/views/view-evolution.html

[6] "Mysql." [Online]. Available: https://www.mysql.com/

[7] "The GNU nano text editor." [Online]. Available: https://www.nano-editor.org
[8] National Institute of Standards and Technology (NIST), "National Vulnerability Database." [Online]. Available: https://nvd.nist.gov
[9] Google, "OSV: a distributed vulnerability database for Open Source." [Online]. Available: https://osv.dev
[10] "IEEE Xplore." [Online]. Available: https://ieeexplore.ieee.org
[11] "Google Scholar." [Online]. Available: https://scholar.google.com/
[12] "Web of science." [Online]. Available: https://clarivate.com/webofsciencegroup/solutions/web-of-science/
[13] A. L. Johnson, "The analysis of binary file security using a hierarchical quality model," 2022. [Online]. Available: https://scholarworks.montana.edu/xmlui/handle/1/16635
[14] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2022. [Online]. Available: https://www.R-project.org/
[15] H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. [Online]. Available: https://ggplot2.tidyverse.org
[16] S. Garnier, N. Ross, R. Rudis, A. P. Camargo, M. Sciaini, and C. Scherer, *viridis - Colorblind-Friendly Color Maps for R*, 2021, r package version 0.6.2. [Online]. Available: https://sjmgarnier.github.io/viridis/
[17] D. Rabosky, M. Grundler, C. Anderson, P. Title, J. Shi, J. Brown, H. Huang, and J. Larson, "BAMMtools: an r package for the analysis of evolutionary dynamics on phylogenetic trees," *Methods in Ecology and Evolution*, vol. 5, pp. 701–707, 2014.
[18] Fraunhofer FKIE, FKIE-CAD, "cwe_checker v0.5," 2021. [Online]. Available: {{https://github.com/fkie-cad/cwe_checker/archive/refs/tags/v0.5.zip}}
[19] ——, "cwe_checker v0.6," 2022. [Online]. Available: {{https://github.com/fkie-cad/cwe_checker/archive/refs/tags/v0.6.zip}}
[20] N. Juristo and A. M. Moreno, *Basics of software engineering experimentation*. Springer Science & Business Media, 2013.