Benchmarking Flaws Undermine Security Research

Erik van der Kouwe*, Gernot Heiser†, Dennis Andriesse*, Herbert Bos* and Cristiano Giuffrida* *Vrije Universiteit Amsterdam, The Netherlands—{e.vander.kouwe,d.a.andriesse,h.j.bos,c.giuffrida}@vu.nl
†UNSW Sydney and CSIRO's Data61—gernot@unsw.edu.au

Abstract—Benchmarking systems is difficult. Mistakes can compromise guarantees and threaten reproducibility and comparability. We conduct a study to show that benchmarking flaws are widespread in systems security defense papers, even at tier-1 venues. We aim to raise awareness and provide recommendations for safeguarding the scientific process in our community.

Index Terms—benchmarking, computer systems, performance evaluation, reproducibility of results, security, standardization guidelines

1 Introduction

Benchmarking is essential in systems security—to compare different solutions and reproduce prior results. At every program committee meeting for every top venue in our field, heated discussions revolve around the question whether the performance numbers reported in papers X and Y are reliable and how they relate to each other. Making the wrong call is bad, as nobody wants to accept or reject papers for the wrong reasons. And after we accept a paper, we want to be able to reproduce and compare the results in a meaningful way. In this article, we examine publications from top security conferences to determine whether they contain *benchmarking flaws* that threaten the validity of their results. As we do not allege intent, we avoid the term *benchmarking crimes*, used earlier in the literature, including Heiser's web page [1], that provides the basis for the flaws/crimes we study.

Bluntly speaking, benchmarking flaws threaten the validity of the research results in publications. The obvious question then is: How safe are we as a community from this threat? And if we are not safe, how serious is this threat, and how can we mitigate it? Phrased differently, we want to know how well the systems security research community detects anomalies in benchmarking in evaluation sections of papers published in tier-1 venues, what the consequences are of false negatives, and how to fix these "vulnerabilities".

In the community, there is wide agreement that performance benchmarks are important to advance the field [2]. In systems, almost all security mechanisms incur some performance overhead [3]. The aim is to keep the overhead as low as possible, while raising the bar for attackers as high as possible. Given an unlimited performance budget, techniques to build secure systems under common threat models are already well-established—memory safety being a typical example [4]. As a result, much modern systems security research focuses on practical defenses [such as control-flow integrity [5] or randomization [6]], that trade off some security to achieve realistic performance guarantees.

In this article, we take a closer look at benchmarking flaws in systems security. While it would be good to also benchmark the security of a solution, doing so in an unbiased way is much harder [7], and this article focuses primarily on performance benchmarking of defenses (expanding on other dimensions when appropriate). After discussing the objectives of performance benchmarking in general, we carefully explore all the pitfalls that authors may encounter when assessing the performance of their research artefact. For each of these benchmarking flaws, we explain the negative impact they may have on the validity or usefulness of the evaluation.

Finally, we assess the state of benchmarking in systems security. We selected systems defense papers from the tier-1 computer security venues where systems security defenses are routinely published (IEEE Symposium on Security and Privacy, USENIX Security, ACM Conference on Computer and Communications Security (CCS), and Network and Distributed System Security Symposium (NDSS)). We sifted through 50 papers and analyzed them for benchmarking flaws. For this purpose, we selected all defense papers with benchmarking results published in 2010 and 2015. As nearly all papers in our data set have at least some benchmarking issue (and many have several) and we found no clear difference between the more recent and the older papers, we conclude that improper benchmarking is a serious threat with little improvement in recent years.

This article summarizes the key results of our study; our earlier conference paper contains the complete results [8]. Here we discussion the implications of these results in more detail, and propose an approach to solve the problem of benchmarking flaws in our community.

2 Benchmarking Flaws

Almost every paper in computer systems requires an evaluation that determines whether and how well the presented system achieves its goals. One important purpose of the evaluation is to compare against other work: it should show that the system improves the state of the art in some way and allows possible subsequent papers to demonstrate that they improve this system. To allow for comparison, an evaluation must meet a number of requirements. First of all, it should be complete, in the sense that it verifies all the claimed

contributions of the system and shows the extent of any negative impact the system may have. All the presented results must be relevant in the sense that they actually tell the reader something meaningful about the system. Another important characteristic is soundness, that is, the requirement that all the numbers measure what is intended with reasonable accuracy and repeatability. Finally, a general principle of science requires papers to be reproducible. That is, the information provided in the paper should be sufficient enough to allow others to build the system and perform its evaluation in the same way as the original. A good paper should meet all of these requirements, but unfortunately, experience shows that this is often hard to come by in practice. Indeed, we found that most papers contain a number of *benchmarking flaws* that violate these properties.

In this section, we describe the benchmarking flaws we identified and explain their importance. Our list is based in large part on a web page by Heiser [1] (who uses the term *benchmarking crimes*), aimed at operating systems researchers. We adapt the list to the context of security research, and also perform a systematic and large-scale survey of systems defense papers at top conferences (see Section 3) to determine whether these benchmarking flaws are common in published systems security papers. We find that these flaws apply not only to the operating systems community, but extend to other subfields of computer systems, in particular, systems security. This is particularly important because, as we shall see, Heiser's original web page [1] published in 2010 had insufficient impact in the systems security community. Benchmarking flaws are still widespread and their relevance has, in fact, grown over time.

We placed the 22 benchmarking flaws we identified into groups and assigned codes (a letter for the group plus a number for the specific flaw) to simplify later references to them. We describe the groups and the individual benchmarking flaws and later elaborate on their impact in Section 3. A more detailed description including examples can be found in the conference paper [8].

2.1 Selective Benchmarking

There is no single number that can fully express how well a system performs. Performance overhead is multidimensional as different operations are affected in different ways. For example, a system that performs CFI [5] instruments indirect branches but leaves other operations alone. Therefore, it is likely to incur substantial overhead for programs and workloads that perform many function calls, especially if they are indirect (e.g., common C++ programs), but it will incur minimal overhead if the program spends most of its time in a loop that calls no functions. This has several implications for benchmarking, and when a paper does not consider these implications it might result in a performance evaluation becoming anywhere from slightly inaccurate to completely meaningless.

- *A1: Not evaluating potential performance degradation* occurs whenever a paper does not include benchmarks that evaluate all the operations whose performance one might reasonably expect to be impacted.
- *A2: Benchmark subsetting without proper justification* applies to papers that arbitrarily select a subset of benchmark suites and present it as a single overall performance overhead number as if it were still representative.
- *A3: Selective data sets that hide deficiencies* arises when papers fail to test operation performance over an appropriate range of settings (for example, core count), which uncovers all-important performance characteristics.

2.2 Improper Handling of Benchmark Results

Our second group of flaws deals with correctly interpreting benchmarking results. Even when running the right benchmarks, the presentation of their results can be misleading if they are processed in incorrect ways. This group contains five flaws related to the incorrect handling of benchmark results:

- *B1: Microbenchmarks representing overall performance* results in misleading results that provide no indication of how fast a system would run in practice.
- *B2: Throughput degraded by x% $\Rightarrow$ overhead is x%* refers to cases where measurements are conducted in such a way that performance overhead is hidden by idle time (the CPU is not fully loaded).
- *B3: Bad math* refers to incorrect computations using overhead numbers; for example, from the use of percentage points to present a difference in overhead.
- *B4: No indication of significance of data* is an issue whenever averaged measurement results are presented without some indication of the amount of variation.
- *B5: Incorrect averaging across benchmark scores* often occurs when authors use the arithmetic mean to average overhead ratios, while only the geometric mean is correct in this case [9].

2.3 Using the Wrong Benchmarks

This group of benchmarking flaws pertains to using the wrong benchmarks and consists of the following three benchmarking flaws:

- *C1: Benchmarking of simplified simulated system* refers to cases where the benchmarks are not run on a real system but rather an emulated version.
- *C2: Inappropriate and misleading benchmarks* refers to the use of benchmarks that are not suitable for measuring the expected overheads.
- *C3: Same dataset for calibration and validation* applies to papers that test a trained system where the data set used for testing overlaps with the training data set.

2.4 Improper Comparison of Benchmarking Results

Raw measurements like runtime or throughput numbers are rarely meaningful in isolation. Instead, they must be interpreted by comparing them to a baseline to determine how much overhead the system incurs and/or to competing systems to determine whether the system can improve their performance. We separated this issue into the following three different benchmarking flaws:

- *D1: No proper baseline* refers to computing overhead compared to an unsuitable baseline.

- *D2: Evaluating only against oneself* refers to cases where authors compare their new system to their own earlier work rather than the state of the art.
- *D3: Unfair benchmarking of competitors* refers to papers that do compare against competitors but do so in an unfair way.

2.5 Benchmarking Omissions

The following group of flaws covers necessary measurements for evaluations that are not yet covered by the other benchmarking flaws:

- *E1: Not all contributions evaluated* refers to cases where a paper claims to achieve a certain goal but does not empirically determine whether this goal has been reached. It is critical that papers verify claims.
- *E2: Measuring only runtime overhead* occurs whenever a system should be expected to impact performance characteristics that are not measured, such as memory usage.
- *E3: False positives/negatives not tested* is an issue for papers that involve classification but do not verify its accuracy.
- *E4: Elements of solution not tested incrementally* refers to papers that test the combined impact of the proposed approach but fail to isolate whether all the proposed elements of the solution provide a useful contribution to the overall result.

2.6 Missing Information

The following final group contains benchmarking flaws where important information has been left out of a paper:

- *F1: Missing platform specification* applies to papers that lack a description of the hardware setup used to perform the experiments.
- *F2: Missing software versions* is similar but refers to the software.
- *F3: Subbenchmarks not listed* applies to papers that run a benchmarking suite but do not present the results of the individual subbenchmarks, just the overall number.
- *F4: Relative numbers only* involves presenting only ratios of overheads without presenting the actual measured numbers.

3 Study Results

To determine the prevalence of the benchmarking flaws discussed in Section 2 and to gain a better idea of what these flaws look like in practice, we investigated 50 papers published at top security venues. We focused our analysis on the traditional "top 4" venues in security: USENIX Security, Security & Privacy, CCS, and NDSS, selecting all systems defense papers from these venues in 2010 and 2015. The conference paper [8] lists all the papers selected for our analysis.

For each of the 50 selected papers and each benchmarking flaw described in Section 2, we determined whether the paper contains a particular flaw. Two people independently categorized each paper for each flaw as correct, flawed, underspecified, or not applicable. In most cases, both readers came to the same conclusions, suggesting that our methodology is reproducible. For papers where there were some disagreements, the readers discussed their assessments to converge on a final classification. This was the case for eight out of the 50 papers (16%). In only two cases did the discussion lead to the addition of a benchmarking flaw initially missed by one of the readers. The remaining disagreements concerned the precise extent of flaws identified by both readers.

Figure 1 shows the number of papers containing each flaw, counting as a single occurrence a paper that contains the same flaw multiple times. In some cases, we were unable to determine whether the methodology in the paper is sound because important elements of the experiments or their analyses were not specified with a sufficient level of detail. We have classified these paper/flaw pairs as underspecified. Note that underspecification is problematic even if the underlying methodology is sound because it hampers reproducibility and makes it harder for later competitors to perform a fair comparison with prior work.

Our results show that benchmarking flaws are a major problem. Over all pairs of paper and their applicable flaw, the flaw either applies or the paper is underspecified with regard to the flaw in 256 out of the 851 cases (30%). However, not all flaws are equally common. The lack of indication of the significance of data and benchmark subsetting without proper justification are by far the most widespread, affecting 80 and 69%, respectively, of the applicable papers we investigated. None of the other flaws affect a majority of the papers but four additional ones affect 40% or more of the papers to which they apply. This shows that several types of benchmarking flaws are widespread even in peer-reviewed papers at the top venues.

Figure 2 shows a histogram of the number of benchmarking flaws (including underspecification) per paper. High-impact flaws are explained in the next section. It is notable that from our sample of 50 papers, we found only one paper without any benchmarking flaws. Flaws are fairly evenly spread between papers, with many papers being very close to the average number of benchmarking flaws per paper (5.0 for all flaws and 1.7 for high-impact flaws). As such, the results would seem to suggest that the problem of benchmarking flaws is not an issue of a few authors and reviewers being particularly careless (or malicious), but rather a community-wide lack of awareness of or attention to these problems. This is further corroborated by the fact that many prevalent benchmarking flaws require very little effort to fix, as detailed later.

4 Impact

We examine the impact of frequent flaws, i.e., those found in at least 10 papers (the conference version [8] has the complete analysis).

4.1 Selective Benchmarking

A1: Not Evaluating Potential Performance Degradation. We found two major groups of papers that contain this flaw: those where overhead figures are missing entirely and those

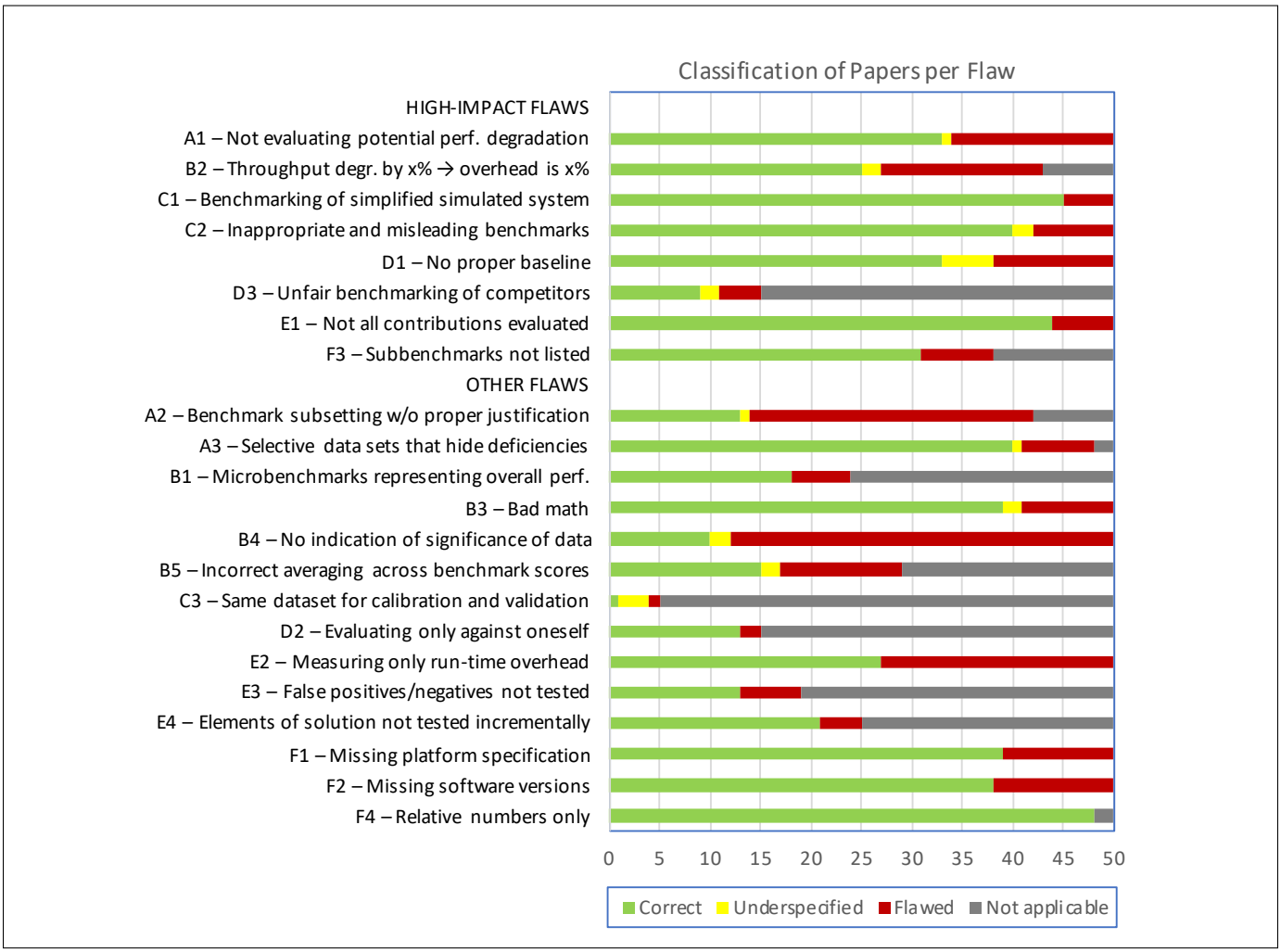


Fig. 1. Benchmarking flaws study overview.

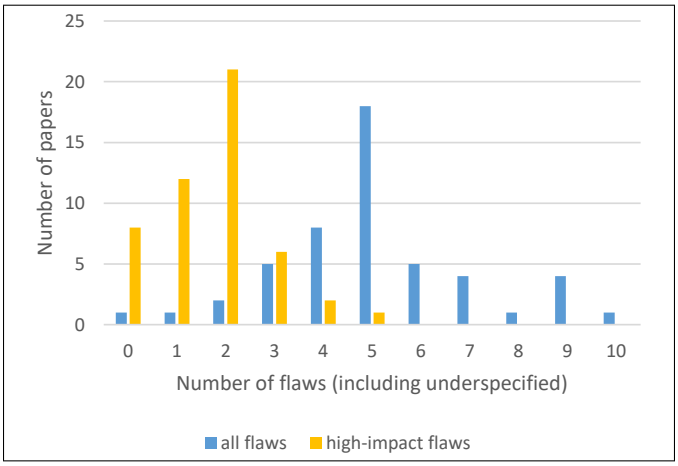


Fig. 2. Histogram of number of flaws per paper

that do not reflect all potential slowdown. In both cases, this flaw makes it difficult (if not impossible) to assess the practicality of the presented solution and its improvements over the state of the art. Moreover, papers that present inappropriate performance measurements may even hamper scientific progress because they prevent competing systems that perform poorly on these inappropriate measures, or not as efficiently on appropriate measures, from being published. Even worse, they may encourage more benchmarking flaws in future systems, as authors struggle to exceed overly optimistic performance figures. As such, we consider this flaw to be **high impact**.

A2: Benchmark Subsetting Without Proper Justification. We found that many papers using standardized benchmarking suites leave out some subbenchmarks. Based on the particular benchmarks that are often left out, it is very likely that this will result in an underestimation of performance overhead in practice [8]. We conclude that leaving out subbenchmarks can have a major impact on the soundness of measurements as well as the comparability between competing systems and therefore requires a proper and exact justification. Moreover, if different papers use different subsets, the overall slowdown is no longer suitable for comparing performance.

4.2 Improper Handling of Benchmark Results

B2: Throughput Degraded by x% ⇒ Overhead Is x%. Based on our study, we believe that all instances of this benchmarking flaw are likely to result in an underestimate of performance overhead, although without the necessary data it is impossible to determine by how much. Because this flaw is likely

to affect the soundness of performance measurements in all cases, we consider it to be a **high-impact** benchmarking flaw.

B4: No Indication of Data Significance. Some indication of variation is important because it is an indication of how reliable the numbers are and whether, given the measurement inaccuracy, the measured differences are actually meaningful. However, we expect the overall impact of this flaw to be relatively mild for papers where researchers set up their experiments correctly.

B5: Incorrect Averaging Across Benchmark Scores. To determine the impact of incorrect averaging, we compared the geometric mean with the presented average. For four out of the eight papers where we could compute this mean, the difference in the arithmetic mean is at least 1% higher than that of the geometric mean, overestimating the overall overhead. For the worst case that we found, the arithmetic mean is more than twice the geometric mean, while the remainder overestimate overhead by 2–16%.

4.3 Improper Comparison of Benchmarking Results

D1: No Proper Baseline. We found that some papers have an incorrect baseline, while others do not present one at all. The former are always likely to either underestimate overhead or overestimate effectiveness. This threatens both the soundness and comparability of the results. Absolute performance numbers without baselines cannot be compared between systems and therefore provide little meaningful information. Because we found that the lack of a proper baseline was a serious problem in all cases, we consider this flaw to be **high impact**.

4.4 Benchmarking Omissions

E2: Measuring Only Runtime Overhead. Papers that do not measure important sources of overhead other than runtime are incomplete; however, the impact of this incompleteness differs from case to case. If, for example, memory overhead can theoretically be assumed to be minor and similar to that of prior work, the impact is limited. If, on the other hand, there is reason to believe that the paper incurs significant memory overhead yet does not measure it, this could be a problem for later papers that improve upon this overhead.

4.5 Missing Information

F1: Missing Platform Specification. This benchmarking flaw makes reproducing the exact results based on the contents of the paper impossible. It may make the results less comparable but does not affect the validity of the results.

F2: Missing Software Versions. This benchmarking flaw hampers reproducibility, as the software about which information is missing should be expected to have an impact on performance.

5 Recommendations

Our analysis shows that benchmarking flaws are very common and potentially have a major impact on the quality of published research in systems security. Based on the results, we suggest that being mindful of the most important flaws will improve the quality of published research with relatively little effort, and we have provided a list of such suggestions [8].

However, we also feel that our results expose an underlying problem: benchmarking practices do not receive the attention and priority they deserve in our community. Making a lasting impact requires more than just pointing authors to a few specific problems, which has been done before (though without examining published papers) by Heiser [1]. Instead, we would like to provide authors, reviewers, and program committee chairs with some guidance as to the *right* way of benchmarking in systems security. This will not be the last word on benchmarking in security, but we hope that our suggestions will begin the discussion.

We do not do research and write papers in isolation, as we are part of a community that sets examples and expectations of what a good paper is supposed to be like. Moreover, there is great pressure to publish, and putting a lot of effort in evaluation is not always the most rewarding way to spend one's limited time. As such, incentive structures matter for benchmarking practices and, as a community, we need to think about how we can best reward good research. To improve our benchmarking practices—as our study has shown is urgently needed—bottom-up changes from individual authors are insufficient; rather, a coordinated effort is needed. In particular, we consider the changes we believe are necessary for the community as a whole, for the individual authors writing papers, and for the program committees that decide which papers to publish or not publish.

5.1 Community

The community of security researchers must be the starting point for any solution to the problem of benchmarking flaws, as any approach without community support will be unrewarding for researchers and therefore unlikely to succeed. There are several tasks that we feel individual community members should address and others that should be addressed by the community as a whole.

While our paper [8] identifies a number of problems in benchmarking and examines the way they are addressed in top conferences, this should not be the last word on the subject. We call upon the community to further investigate these issues because it is impossible to come up with the best solution without first knowing the full extent of the problem. The following are a several specific directions that we feel must be explored:

1) We only cover systems defenses, but we expect that similar issues exist in other areas of systems security that rely on experimental validation.
2) We focus primarily on performance, but the measurement of effectiveness (that is, the level of security) is also very important and even harder to measure properly than is performance.
3) Even though we include all four tier-1 conferences in our field, it would be beneficial to also consider lower-tier conferences where many in our field also publish, and where the review process may be simplified and/or the program committee members less experienced.

4) Although we take a bird's-eye view and look for many flaws in a large number of papers in a binary fashion, there is value in more in-depth studies of why and how specific flaws are introduced and what their impact in practice is (for example, by reproducing experiments using different setups).
5) In cooperation with conference organizers, it would be possible to compare submitted, accepted, and camera-ready papers, which would be valuable to evaluate the effectiveness of the review process to reduce the number of benchmarking flaws.

Although conferences focus mostly on technical content, we feel it is important to also use these venues to draw attention to research practices within our community. A practical solution may be to set up workshops cohosted with our top conferences that serve specifically as a place to discuss this type of metaresearch.

We call upon the community to establish a consensus on a set of best practices. Leading researchers in each subfield, working together, could establish a set of accepted benchmarks and write a performance evaluation guide. In particular, this evaluation guide should include a checklist that authors and program committees can apply to determine whether a certain paper meets the minimum requirements. Our list of benchmarking flaws can serve as a basis, though we would recommend a positively formulated checklist that indicates what we expect of a paper rather than one that focuses only on specific ways to get it wrong. This checklist must receive broad endorsement in the community before it can be effective.

More concretely, we should agree upon the benchmarks to use (as a minimum) for which types of systems and how they are to be configured. This can help new researchers learn how to properly perform benchmarking, set standards to ensure that systems are benchmarked properly, and make certain that the performance measurements of similar systems are comparable. Even though some benchmarks are already in widespread use in our community [for example, Standard Performance Evaluation Corporation (SPEC) CPU], it is important to highlight their constraints (in this case, it only exercises the CPU and mostly remains in usermode) and provide additions and/or alternatives where they are not suitable. Configuration is also of particular importance because, in many cases (for example sufficient concurrency for ApacheBench, or a suitable optimization level for SPEC CPU), it has a significant impact on the soundness or the result.

In addition to guidelines that indicate what is expected of authors, we feel that the community should also actively help authors achieve these goals with reasonable effort. In particular, it would be helpful to build open source frameworks that save researchers time when setting up benchmarking for their systems in accordance with the guidelines. As for the investigations of existing benchmarking practices, we need suitable venues to publish this article, not only to make it more worthwhile for community members to make the effort to build tools that benefit us all, but also to ensure visibility and community involvement in these projects. Practically speaking, these could be the same workshops that we also proposed for publication of metaresearch on benchmarking practices in our field.

5.2 Authors

Improving benchmarking practices in systems security critically depends on the authors who actually run those benchmarks and present their results. We hope that the community efforts described in the previous section will make clear to authors what is expected of them and provide them with the information and tools that they need to achieve this with minimal effort. In this section, we describe how we expect authors to follow best practices and how they should be able to use resources to the fullest extent possible.

Our first recommendation is that authors start considering benchmarking requirements early in the project, rather than as a part of the final paper writing stage. Evaluation is fundamentally a part of research, and research prototypes should be designed to facilitate easy (and, if possible, automated) verification of each individual contribution claimed in the work. As such, listing contributions should be one of the first steps and will serve as a basis to select the most suitable benchmarks based on the community consensus described in the previous section. In addition, nearly all of the proposed systems involve some kind of tradeoff. For example, security benefits will often result in slower performance and higher memory usage. When listing contributions, authors should be precise about the limitations and cost and evaluate these with suitable benchmarks, again, based on community consensus.

As for the paper-writing stage, our study shows that in many cases, critical information is missing, which, if explicitly added to the paper, could have prevented some of the benchmarking flaws. Our advice is to be complete and clearly list the limitations, explaining why these limitations are there. In particular, for cases where standard solutions are not viable, the author should clearly explain why this is the case and select the next best alternative. The author should discuss unambiguously what the impact of these differences is expected to be. When presenting evaluation results, authors must ensure that they present the most meaningful quantities (avoiding flaws B3 and B5) and include relevant statistics to indicate how statistically significant the measured impacts are (avoiding flaw B4). With regard to completeness, authors should take utmost care to describe all parts of their experimental setup to allow for independent verification of their approach. Ideally, authors should open source their code and the environment/scripts needed for reproduction. By following these steps, many flaws can be prevented with relatively little effort.

Finally, researchers should apply the community consensus checklist to determine whether they meet the requirements and have avoided benchmarking flaws. Even when carefully planning experiments and writing the paper, it is easy to forget about some of the points. The checklist offers an opportunity to match the benchmarking in the paper with the reviewers' expectations.

5.3 Program Committees

The goal of any researcher is to have their work published and, within our community, conferences are the main venue to achieve this. Program committees act as gatekeepers,

allowing only the work that is of sufficient quality to be published. Although traditionally program committees could only decide to accept or reject a paper, the IEEE Symposium on Security and Privacy has switched to a rolling deadline system, with other top security conferences recently following suit. This introduces the possibility to require revisions, and allows the program committee to give a paper that is interesting but not of sufficient quality an opportunity to improve, followed by a second round of reviews, similar to the model used by scientific journals.

We propose using this opportunity to enforce the guidelines to be formulated by the community. In particular, reviewers should consider the benchmarking practices checklist and mark any benchmarking flaws found in their reviews. Benchmarking flaws need not lead to immediate rejection but should be fixed in the revision stage. In many cases, it would likely suffice to require that any deviation from community best practices be made clear and justified in the paper. In more severe cases, where it appears that high-impact benchmarking flaws do, in fact, undermine the validity of the results and a round of revisions is not sufficient to address the problem, we should reject those papers. It is important to have specific discussions about how we weigh correctness against expected impact. With benchmarking issues exposed, it becomes easier to make this tradeoff and reduce the number of papers that do not meet the community quality standards.

6 Related Work

Benchmarking in Systems Security: Although there have been several studies that determine whether computer science papers perform measurements in appropriate ways (among others [10], [11], [12], [13]), to the best of our knowledge, none of them are specific to benchmarking in systems security. The most closely related work is Heiser's original web page about benchmarking crimes [1], which serves as inspiration for this article. Compared to Heiser's web page (which has been updated since our publication), we propose an extended classification and present a systematic analysis of benchmarking flaws in peer-reviewed defense papers. We also formulate concrete recommendations.

Studies Considering Evaluation Quality: A number of authors have examined how well papers in various fields evaluate their work. Kuz et al. [10] investigate benchmarking for multicore systems to propose a better approach but only consider six papers. Skadron et al. [11] investigate papers in computer architecture to determine their topics and performance evaluation techniques. They provide an overview and discussion of the various techniques but do not go into any depth about incorrect benchmarking practices. Mytkowicz et al. [12] present a study to determine whether measurement error is considered correctly in computer systems experiments and provide suggestions on how to improve this. Rossow et al. [13] study the methodological rigor and prudence in papers using malware execution. Although their approach for identifying flaws is similar to ours, the pitfalls they identify are quite different because they focus on malware analysis rather than performance.

Benchmarking Advice: Some other papers also provide benchmarking advice but do so without a systematic study, instead using examples and their own tests to verify that the identified pitfalls result in questionable results. Schwarzkopf et al. [14] identify benchmarking problems in cloud research and Seltzer et al. [15] discuss the problems associated with standardized benchmarks in file systems research. Even though these studies demonstrate important benchmarking problems, the lack of a systematic study means they cannot determine the impact these potential problems have on the research literature.

7 Conclusion

Although the security community expends considerable effort on defending systems from increasingly dangerous threats, it devotes much less attention to the correctness of research results. Benchmarking flaws, in particular, have been largely neglected. As the focus of systems research is increasingly shifting to practical, low-overhead defenses, benchmarking flaws are increasingly relevant and are now the "elephant in the room." We assessed the magnitude of the problem in 50 defense papers in top systems security venues, suggesting that benchmarking flaws are widespread and show no sign of improvement, thus hampering research comparability and reproducibility. Encouragingly, many common benchmarking flaws can be easily prevented by following our guidelines for authors. We have made available a checklist [16] to allow interested readers to easily apply our findings to their own work.

Acknowledgments

This project was supported by the European Union's Horizon 2020 research and innovation program under grant agreement 786669 (ReAct) and 825377 (UNICORE), by the U.S. Office of Naval Research under contract N00014-17-1-2782, by Cisco Systems, Inc. through grant 1138109, and by The Netherlands Organisation for Scientific Research (NWO) 639.023.309 VICI "Dowsing" and NWO 639.021.753 VENI "PantaRhei". This article reflects only the authors' view. The funding agencies are not responsible for any use that may be made of the content.

References

[1] G. Heiser, "Systems benchmarking crimes," https://www.cse.unsw.edu.au/~gernot/benchmarking-crimes.html.
[2] S. E. Sim, S. Easterbrook, and R. C. Holt, "Using benchmarking to advance research: A challenge to software engineering," in *ICSE*, 2003.
[3] J. Wagner, V. Kuznetsov, G. Candea, and J. Kinder, "High system-code security with low overhead," in *IEEE Security&Privacy*, 2015.
[4] S. Nagarakatte, J. Zhao, M. M. Martin, and S. Zdancewic, "Softbound: Highly compatible and complete spatial memory safety for C," in *PLDI*, 2009.
[5] M. Abadi, M. Budiu, U. Erlingsson, and J. Ligatti, "Control-flow integrity," in *ACM CCS*, 2005.
[6] S. Crane, C. Liebchen, A. Homescu, L. Davi, P. Larsen, A.-R. Sadeghi, S. Brunthaler, and M. Franz, "Readactor: Practical code randomization resilient to memory disclosure," in *IEEE Security&Privacy*, 2015.
[7] S. M. Bellovin, "On the brittleness of software and the infeasibility of security metrics," *IEEE Security&Privacy*, 2006.
[8] E. van der Kouwe, G. Heiser, D. Andriesse, H. Bos, and C. Giuffrida, "Sok: Benchmarking flaws in systems security," *Proc. 4th IEEE Eurpean Sym. Security and Privacy (EuroS&P)*, 2019.

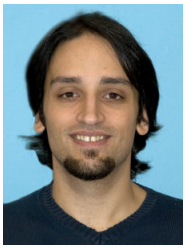
[9] P. J. Fleming and J. J. Wallace, "How not to lie with statistics: the correct way to summarize benchmark results," *Communications of the ACM*, vol. 29, no. 3, pp. 218–221, 1986.
[10] I. Kuz, Z. R. Anderson, P. Shinde, and T. Roscoe, "Multicore OS benchmarks: We can do better," in *HotOS*, 2011.
[11] K. Skadron, M. Martonosi, D. August, M. Hill, D. Lilja, and V. S. Pai, "Challenges in computer architecture evaluation," *IEEE Computer*, vol. 36, no. 8, pp. 30–36, 2003.
[12] T. Mytkowicz, A. Diwan, M. Hauswirth, and P. F. Sweeney, "Producing wrong data without doing anything obviously wrong!" *ACM SIGPLAN Notices*, vol. 44, no. 3, pp. 265–276, 2009.
[13] C. Rossow, C. J. Dietrich, C. Grier, C. Kreibich, V. Paxson, N. Pohlmann, H. Bos, and M. Van Steen, "Prudent practices for designing malware experiments: Status quo and outlook," in *IEEE Security&Privacy*, 2012.
[14] M. Schwarzkopf, D. G. Murray, and S. Hand, "The seven deadly sins of cloud computing research," in *HotCloud*, 2012.
[15] M. Seltzer, D. Krinsky, K. Smith, and X. Zhang, "The case for application-specific benchmarking," in *HotOS*, 1999.
[16] "Threats to validity and relevance in security research," https://www.vusec.net/threats-to-validity-in-security-research/.

Erik van der Kouwe Erik van der Kouwe is an assistant professor at the VUSec systems security group at the Vrije Universiteit Amsterdam. He received his PhD in software reliability from prof. Andy Tanenbaum, and afterwards broadened hist scope to include systems security. His recent work is on practical compiler-assisted defenses against zero-day vulnerabilities and on properly benchmarking such defenses.

Gernot Heiser Gernot Heiser is Scientia Professor and John Lions Chair of Operating Systems at UNSW Sydney, and Chief Research Scientist at CSIRO's Data61. His research is on high-performance, highly dependable operating systems, especially microkernels, for security- and safety-critical systems, with a strong track record of transferring research outcomes to real-world applications. He holds a PhD from ETH Zurich, an MSc from Brock University and a BSc from the University of Freiburg. He is a Fellow of the IEEE, a Fellow of the ACM and a Fellow of the Australian Academy of Technology and Engineering (ATSE).

Dennis Andriesse Dennis Andriesse is a security researcher at Intel. He obtained his PhD in System and Network Security from Vrije Universiteit Amsterdam and has a broad interest in low-level security.

Herbert Bos Herbert Bos is full professor of Systems Security at Vrije Universiteit Amsterdam where he co-leads the VUSec group with Cristiano Giuffrida and Erik van der Kouwe. He obtained his Ph.D. from Cambridge University Computer Laboratory (UK). Coming from a systems background, he drifted into security a few years ago and never left. He is doomed to wander the earth and not find true happiness until he has recruited an additional assistant professor in systems security for VUSec. So if this is you, please apply!



Cristiano Giuffrida Cristiano Giuffrida is an Assistant Professor in the Computer Systems Section at the Vrije Universiteit Amsterdam. His research interests include systems security, reliability, and availability. Giuffrida received a PhD from the Vrije Universiteit Amsterdam in 2014. He was awarded the Roger Needham Award and the Dennis M. Ritchie Award for the best PhD thesis in Computer Systems (Europe and worldwide) in 2015.