How do Compiler Bugs Affect OS Kernels

Huazhao Chen
lyican53@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Hao Zhong
zhonghao@sjtu.edu.cn
Shanghai Jiao Tong University
Shanghai, China

Abstract

Compiler bugs can silently introduce bugs into their compiled code. Although compiler bugs are critical in the development of OS kernels, most studies on compiler bugs analyze the characteristics of only compiler bugs themselves. Their findings are less interesting to outsiders, *e.g.*, the OS community. Although two recent papers have started to analyze how compiler bugs affect the development of other projects, no study has been conducted to explore how such bugs affect OS kernels. As a result, many questions are still open. For instance, in the development of OS kernels, are most compiler bugs identified by OS programmers themselves? When bypassing compiler bugs, what is the relationship between compiler-bug symptoms and modified OS components? The answers to the above questions are useful for both compiler and OS programmers. To answer the above questions, we conducted the first empirical study on how compiler bugs affect OS kernels. We collected 494 workarounds of compiler bugs from the revision histories of 7 popular OS kernels. We analyze these workarounds and explore the answers to four research questions. We have summarized some findings according to our early result. For instance, we found that in most workarounds, compiler bugs are not reported by OS programmers.

ACM Reference Format:
Huazhao Chen and Hao Zhong. 2025. How do Compiler Bugs Affect OS Kernels. In *Proceedings of The 48th ACM SIGSOFT International Conference on Software Engineering (ICSE 2026).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Software infrastructures are the foundation of application development. For instance, as a typical software infrastructure, operating systems (OSes) provide programmers with numerous application programming interfaces (APIs) for managing hardware and software resources, *e.g.*, memory, and devices [1]. According to a recent study [2], a project is likely to trigger compiler bugs if it has more than a half million lines of code. As OSes have million lines of code, they can trigger compiler bugs, and such bugs can further introduce errors into compiled OSes. As OS bugs can affect many applications, understanding the affection and effects of compiler bugs in OS kernels is critical for both OS and compiler programmers.

ICSE 2026, Brazil

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Table 1: Dataset

Kernel	Workaround	Report	Link	MLOC
Linux	340	231	367	28.29
freebsd	109	53	115	21.44
Serenity	13	14	14	1.15
ReactOS	11	7	13	8.77
openbsd	10	9	10	22.56
zephyr	9	4	9	2.67
AROS	2	2	2	2.67
Total	494	320	530	87.55

[1] MLOC represents million lines of code.
[2] Link represents the number of links between one compiler bug report and one workaround.

Due to the importance of compiler bugs, researchers [3–5] conducted various empirical studies to understand compiler bugs. Although these findings are useful, their audience is narrow. Recently, researchers [2, 6] have started to analyze the far-reaching impacts of compiler bugs. These studies are beneficial for both compilers and other projects. Although their findings are beneficial, they do not analyze how compiler bugs affect the development of OS kernels. OS kernels have millions of lines of code and heavily use the asm volatile. In addition, their source files are compiled with unusual flags, *e.g.*, `-nostdlib`. This flag eliminates the buffering layer of standard libraries and the kernel startup code, causing the kernel code to be more directly exposed to the low-level compiler behavior and bugs. As a result, compiler bugs can have a deeper impact on OS kernels. If compiler bugs introduce bugs into OS kernels, they can affect many applications. As the prior studies do not focus on OS kernels, many questions in this regard are still open.

To deepen the knowledge about how compiler bugs affect OS kernels, we explore the following questions:

- **RQ1: Who reports the compiler bugs?** The answers are useful for understanding the roles of compiler and OS kernel programmers when bypassing compiler bugs.
- **RQ2: How long does it take from reporting bug reports to bypassing them?** The answers are useful for understanding to what degree compiler bugs affect OS kernels.

2 Methodology

Dataset. Table 1 shows the dataset of our study. We collected 494 different workarounds that mention compiler bugs from 7 popular OS kernels. Almost all collected workarounds are related to `gcc` and `clang`. And three workaround records from ReactOS are related to the `msvc` compiler. Following the pioneer studies [2, 6], we used the URLs of compiler bug reports as keywords to query commits from the revision histories of OS kernels to find workarounds of compiler bugs. If a commit message contains these URLs, it should

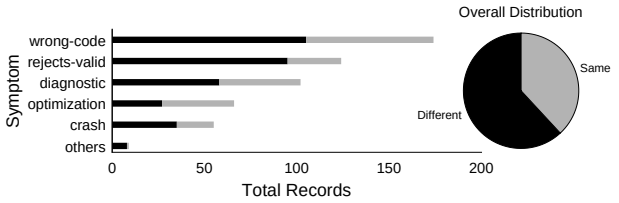

Figure 1: Distribution of knowledge flow

be a workarounds that bypass compiler bugs. Our tool retrieved 496 workarounds. After manual inspection, we removed two irrelevant workarounds. A compiler bug can be mentioned in more than one workaround, and a workaround can also mention more than one compiler bugs. As a result, we collected more workarounds than bug reports and collected more links than the maximum value of compiler bugs and workarounds.

General Protocol. For RQ1, we investigate knowledge flow directions between compilers and OS kernels. We implement a tool to extract the authors of workarounds and the reporters of compiler bugs. We determine that the author of a workaround is the reporter of a compiler bug if their names are identical. The symptoms of compiler bugs can affect the direction of the knowledge flow. He and Zhong [2] merged the categories of `gcc` and `clang` and built a new taxonomy of compiler bugs. Following their taxonomy, we group the knowledge flow directions by symptoms.

For RQ2, we calculate the time interval between the commit time of an OS-kernel workaround and the reporting time of the corresponding compiler bug that the workaround mentions. Meanwhile, we calculate the difference between the two knowledge flow directions and use the cumulative distribution function (CDF) to show the distribution under different directions.

3 Early Results

RQ1: Knowledge Flow. Figure 1 presents the results. We find that in 61.9% of the links, compiler bugs are not reported by programmers of OS workarounds. It indicates that in most cases, OS kernel developers identify the impact of known compiler bugs and implement workarounds to bypass them. The bug knowledge is mainly from compiler to OS kernels. We also notice that wrong-code bugs are the most bypassed, accounting for 32.8% of the total bugs. Compared with the 23.29% wrong-code bugs in general projects analyzed by He and Zhong [2], we find that more wrong-code bugs affect OS kernels. In addition, only in optimization bugs, the knowledge flow from OS kernels to compilers is more common than the reverse direction. This indicates that OS developers identify more optimization bugs than other symptoms bugs by themselves.

RQ2: Time Interval. Figure 2 shows the results. The horizontal axes denote the time intervals in days. The vertical axes denote the CDFs. We can read that when the bug reporter and workaround author are the same,70% of workarounds can be implemented within 100 days after the bug report. When the bug reporter is different from the workaround author, workarounds that can be implemented within 100 days of the bug report account for only 30%. Combine the observation with the median delay of different situations in figure, we find that when compiler bugs are identified by other

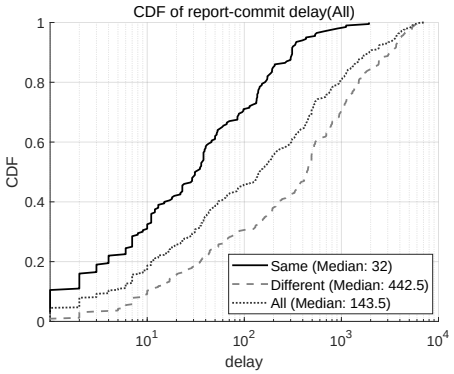

Figure 2: CDF from reporting to bypassing compiler bugs

programmers, their impact and handling time is much longer. This is due to for compiler bugs reported by others, it is costly for kernel developers to notice and identify whether they actually cause an impact. The early results of RQ1 indicate that this situation is more common. Therefore, the answers to these two questions can encourage compiler bug reporters or others to recommend potentially harmful bugs to the OS kernel, thereby reducing the time kernel developers spend noticing and identifying these impacts.

4 Work Plan

To extend this work to a full paper, we plan to explore more RQs:

RQ3: How similar are the code samples and the real code? The compiler bug reports have their own code samples. An interesting question is whether these code samples are helpful to identify the impact in OS kernels? To answer this question, we plan to calculate the similarity of code samples in compiler bug reports and the source code changed before the corresponding workarounds. The answers are useful for OS programmers to identify and reduce the triggering code of compiler bugs in OS kernels.

RQ4: How do bug symptoms relate to modified OS components and workaround size? Compiler bugs have different symptoms and OS kernels also have different components. A question worth exploring is what is the association between bug symptoms and OS components, and how do they affect the modified code lines of corresponding workarounds? We plan to categorize the impacted OS components based on the files modified in the workarounds and explore the relationship between symptoms, components and the number of modifier code lines. The answers are useful for OS and compiler programmers to allocate the resources to handle the impact of compiler bugs or to fix these bugs.

References

[1] R. E. Bryant and D. R. O'Hallaron. *Computer systems: a programmer's perspective*. Prentice Hall, 2011.
[2] Z. He and H. Zhong. From bug report to workrounds:the real-world impact of compiler bugs. In *Proc. SANER*, page to appear, 2025.
[3] A. Romano, X. Liu, Y. Kwon, and W. Wang. An empirical study of bugs in webassembly compilers. In *Porc. ASE*, pages 42–54, 2021.
[4] Q. Shen, H. Ma, J. Chen, Y. Tian, S.-C. Cheung, and X. Chen. A comprehensive study of deep learning compiler bugs. In *Proc. FSE*, pages 968–980, 2021.
[5] Z. Wang, D. Bu, A. Sun, S. Gou, Y. Wang, and L. Chen. An empirical study on bugs in python interpreters. *Transactions on Reliability*, 71(2):716–734, 2022.
[6] H. Zhong. Understanding compiler bugs in real development. In *Proc. ICSE*, page to appear, 2025.