Flexible and Concise Spectre Mitigations for BPF

FGBS-Spring 2023 Presentation Abstract

Luis Gerhorst
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
Ruhr-Universität Bochum (RUB)
gerhorst@cs.fau.de

Henriette Hofmeier
Ruhr-Universität Bochum (RUB)
henriette.hofmeier@rub.de

Timo Hönig
Ruhr-Universität Bochum (RUB)
timo.hoenig@rub.de

1 MOTIVATION

Operating systems rely on system calls to allow the controlled communication of isolated processes with the kernel and other processes. Every system call includes a processor mode switch from the unprivileged user mode to the privileged kernel mode. Although processor mode switches are the essential isolation mechanism to guarantee the system's integrity, they induce direct and indirect performance costs as they invalidate parts of the processor state. In recent years, high-performance networks and storage hardware has made the user/kernel transition overhead the bottleneck for IO-heavy applications. To make matters worse, security vulnerabilities in modern processors (e.g., Meltdown) have prompted kernel mitigations that further increase the transition overhead.

2 BACKGROUND

Linux's extended Berkeley Packet Filter (BPF) allow unprivileged user processes to load safety-checked bytecode into the kernel. The code is just-in-time compiled and executes at near-native speed. Invoking BPF programs in the kernel and calling kernel functions from within BPF is much faster than the respective switch to/from user context [4].

3 PROBLEM STATEMENT

To isolate the BPF programs from the kernel, the bytecode has to be statically verified for memory- and type-safety. Initially, only BPF program paths that could actually execute architecturally were considered during this static analysis. In response to the Spectre vulnerabilities disclosed in 2018 [11, 1, 8, 9, 2], kernel developers have extended the BPF verifier to prevent side-channel leaks from speculatively-executed BPF program paths. This includes (a) inserting instructions to make speculation safe (e.g., index masking), (b) inserting instructions to prevent speculation (e.g., x86 `lfence`), or (c) statically analyzing the behavior on speculative code paths to ensure they are safe. In summary, these mitigations limit the performance and expressiveness of unprivileged BPF programs in order make them safe[1] even in the face of the Spectre attacks.

4 APPROACH

In this talk, we show that there is significant potential to improve the performance and expressiveness of unprivileged BPF without sacrificing safety [3, 12, 10, 6]. Regarding performance, this includes preliminary microbenchmarks of the current Linux v6.1 BPF Spectre mitigations. We show that, depending on the system's context, there are multiple feasible policies to be implemented by the verifier. To solve this, we propose to make the security policy implemented by the BPF verifier configurable at runtime, for example, by integrating instructions to inhibit speculation into the BPF bytecode instruction set [7].

[1]Not considering programming errors such as [5].

REFERENCES

[1] Atri Bhattacharyya, Alexandra Sandulescu, Matthias Neugschwandtner, Alessandro Sorniotti, Babak Falsafi, Mathias Payer, and Anil Kurmus. 2019. SMoTherSpectre. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*. event-place: London United Kingdom. ACM, New York, NY, USA, (November 2019). ISBN: 978-1-4503-6747-9. DOI: 10.1145/3319535.3363194. http://dx.doi.org/10.1145/3319535.3363194.

[2] Claudio Canella, Jo Van Bulck, Michael Schwarz, Moritz Lipp, Benjamin Von Berg, Philipp Ortner, Frank Piessens, Dmitry Evtyushkin, and Daniel Gruss. 2019. A systematic evaluation of transient execution attacks and defenses. In *28th USENIX Security Symposium (USENIX Security 19)*, 249–266. https://www.usenix.org/conference/usenixsecurity19/presentation/canella.

[3] Sunjay Cauligi, Craig Disselkoen, Daniel Moghimi, Gilles Barthe, and Deian Stefan. 2021. SoK: Practical foundations for software Spectre defenses, (May 2021). _eprint: 2105.05801. http://arxiv.org/abs/2105.05801.

[4] Luis Gerhorst, Benedict Herzog, Stefan Reif, Wolfgang Schröder-Preikschat, and Timo Hönig. 2021. AnyCall: Fast and Flexible System-Call Aggregation. en. In *Proceedings of the 11th Workshop on Programming Languages and Operating Systems*. ACM, Virtual Event Germany, (October 2021), 1–8. ISBN: 978-1-4503-8707-1. DOI: 10.1145/3477113.3487267. Retrieved 01/30/2023 from https://dl.acm.org/doi/10.1145/3477113.3487267.

[5] Luis Gerhorst, Henriette Hofmeier, and Daniel Borkmann. 2023. bpf: Fix pointer-leak due to insufficient speculative store bypass mitigation. (January 2023). Retrieved 01/22/2023 from https://git.kernel.org/pub/scm/linux/kernel/git/bpf/bpf.git/commit/?id=e4f4db47794c9f474b184ee1418f42e6a07412b6.

[6] Marco Guarnieri, Boris Köpf, José F Morales, Jan Reineke, and Andrés Sánchez. 2020. Spectector: Principled Detection of Speculative Information Flows. In *2020 IEEE Symposium on Security and Privacy (SP)*. ISSN: 2375-1207. (May 2020), 1–19. DOI: 10.1109/SP40000.2020.00011. http://dx.doi.org/10.1109/SP40000.2020.00011.

[7] Devin Jeanpierre and Chandler Carruth. 2020. Mitigating Spectre v1 Attacks in C++. (January 2020). https://www.open-std.org/jtc1/sc22/wg21/docs/papers/2020/p0928r1.pdf.

[8] Andrea Mambretti, Matthias Neugschwandtner, Alessandro Sorniotti, Engin Kirda, William Robertson, and Anil Kurmus. 2019. Speculator: a tool to analyze speculative execution attacks and mitigations. In *Proceedings of the 35th Annual Computer Security Applications Conference* (ACSAC '19). event-place: San Juan, Puerto Rico, USA. Association for Computing Machinery, New York, NY, USA, (December 2019), 747–761. ISBN: 978-1-4503-7628-0. DOI: 10.1145/3359789.3359837. https://doi.org/10.1145/3359789.3359837.

[9] Ross Mcilroy, Jaroslav Sevcik, Tobias Tebbi, Ben L Titzer, and Toon Verwaest. 2019. Spectre is here to stay: An analysis of side-channels and speculative execution, (February 2019). _eprint: 1902.05178. http://arxiv.org/abs/1902.05178.

[10] Marco Patrignani and Marco Guarnieri. 2021. Exorcising spectres with secure compilers. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. event-place: Virtual Event Republic of Korea. ACM, New York, NY, USA, (November 2021). DOI: 10.1145/3460120.3484534. http://dx.doi.org/10.1145/3460120.3484534.

[11] Michael Schwarz, Martin Schwarzl, Moritz Lipp, and Daniel Gruss. 2018. NetSpectre: Read arbitrary memory over network, (July 2018). _eprint: 1807.10535. http://arxiv.org/abs/1807.10535.

[12] Marco Vassena, Craig Disselkoen, Klaus von Gleissenthall, Sunjay Cauligi, Rami Gökhan Kıcı, Ranjit Jhala, Dean Tullsen, and Deian Stefan. 2021. Automatically eliminating speculative leaks from cryptographic code with blade. en. *Proceedings of the ACM on Programming Languages*, 5, POPL, (January 2021), 1–30. Publisher: Association for Computing Machinery (ACM). ISSN: 2475-1421. DOI: 10.1145/3434330. http://dx.doi.org/10.1145/3434330.