Compiler-based Transformations for Mitigating Timing Side Channel Vulnerabilities

Chanhee Cho (chanheec), Henry Jung (hyesungj)

March 17, 2023

Project Information

Group Information

- Chanhee Cho (chanheec@andrew.cmu.edu)
- Henry Jung (hyesungj@andrew.cmu.edu)

URL for Project Web Page

https://chanhee-cho.github.io

Project Description

In this project, we will explore compiler-assisted mitigations for timing side channel vulnerabilities. The research question we explore is 1) how these mitigations can be implemented as an LLVM pass, which can help incorporate migitations easily, and 2) how heuristics for reducing the overhead of mitigations behave. We will evaluate the effectiveness of these transformations by measuring timing differences over a set of inputs for our microbenchmarks. Moreover, we will measure the performance overhead of these mitigations and the reduced overhead when we implement a heuristic described in one of the literature.
For implementation, we base our initial approach on [1]. We outline our project goals below:

- 75% goal is to implement parts of commonly used control-flow transformations to mitigate timing side channel in LLVM.
- 100% goal is to implement compiler-based transformations that can partly mitigate side channel attacks from data dependencies from memory access.
- 125% goal is to implement a heuristic described in one of the literature to reduce overhead induced by our transformations above.

Plan of Attack and Schedule

We provide a rough timeline for this project below:

3/20-3/24: Familiarize with timing side channels and transformations

3/27-3/31: Start implementing initial transformation on test programs.

4/3-4/7: Implement control flow transformations to mitigate side channels.

4/10-4/14: Start implementing other compiler-based transformations (e.g. data flow transformations).

4/17-4/21: Perform evaluations with experiments (measuring runtime with the implemented transformations). Look into heuristics to reduce the performance overhead.

4/24-4/27: Write report and prepare poster for presenting.

The critical path in the schedule will be the implementation of transformations being effective against mitigating side channels. Work division will be distributed equally.

Milestone

By the milestone, we plan to have achieved our 75% goal of implementing control-flow transformations. We will likely be in progress of implementing other compiler-based transformations (100% goal) by this time.

Literature Search

We have looked into literature for timing side channel attacks and related compiler-based mitigations. Papers we explored can be found in the References section.

Resources Needed

We will be using LLVM for implementing the compiler transformations. For hardware, no specific setup is expected.

Getting Started

We have done some initial literature review on the project topic. We will start by implementing a basic constant-time programming transformation with a simple

if-else program.

References

[1] J. Van Cleemput, B. De Sutter and K. De Bosschere, "Adaptive Compiler Strategies for Mitigating Timing Side Channel Attacks," in IEEE Transactions on Dependable and Secure Computing, vol. 17, no. 1, pp. 35-49, 1 Jan.-Feb. 2020, doi: 10.1109/TDSC.2017.2729549.

[2] B. Coppens, I. Verbauwhede, K. De Bosschere and B. De Sutter, "Practical Mitigations for Timing-Based Side-Channel Attacks on Modern x86 Processors," 2009 30th IEEE Symposium on Security and Privacy, Oakland, CA, USA, 2009, pp. 45-60, doi: 10.1109/SP.2009.19.

[3] Jeroen V. Cleemput, Bart Coppens, and Bjorn De Sutter. 2012. Compiler mitigations for time attacks on modern x86 processors. ACM Trans. Archit. Code Optim. 8, 4, Article 23 (January 2012), 20 pages. https://doi.org/10.1145/2086696.2086702

[4] T. Brennan, N. Rosner and T. Bultan, "JIT Leaks: Inducing Timing Side Channels through Just-In-Time Compilation," 2020 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2020, pp. 1207-1222, doi: 10.1109/SP40000.2020.00007.

[5] Brennan, T. (2020). Static and Dynamic Side Channels in Software. UC Santa Barbara. ProQuest ID: Brennan_ucsb_0035D_14894. Merritt ID: ark:/13030/m5sj70gz. Retrieved from https://escholarship.org/uc/item/7h94v3b3

[6] Qi Qin, JulianAndres JiYang, Fu Song, Taolue Chen, and Xinyu Xing. 2022. DeJITLeak: eliminating JIT-induced timing side-channel leaks. In Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2022). Association for Computing Machinery, New York, NY, USA, 872–884. https://doi.org/10.1145/3540250.3549150