COMPX304-23A: Assignment 4

Individual Assignment
Submit on Moodle
Deadline: 23:00, Friday, 9th May, 2023 Weight: 12.5%

Side Channel Attacks

In our lectures, we discussed how the CPU cache can speed up memory references. However, we also discussed how Spectre attacks can exploit the system by mapping the value of a sensitive piece of information to whether or not a specific cache line exists on the cache.

For this assignment, you are going to conduct a side channel attack to infer the size of the Last Level Cache (LLC)[1] of a system by timing memory accesses. Such a side-channel algorithm will exploit similar vulnerabilities as those used by Spectre and Meltdown.

On Linux, you can use `lscpu`, to get this information. This information will help you design your algorithm and verify your results. Also keep in mind that most intel and AMD CPUs have a cache line of 64 bytes[2].

The overall idea would be to figure out how much memory can be maximally maintained in the system by timing memory reads/writes and comparing fast (in the cache) with slow (not in the cache) accesses. Remember that, once the size of the LLC is crossed, the system will start evicting existing memory objects from the cache; thus, if you try to access them again, that would be a cache-miss, i.e, a slow memory access.

Check this article for some inspiration[3] . One way to perform this experiment/attack is by allocating a very large array. Then, you can access data within it to cause it to load in cache/memory.

Experiment with a variety of configurations and repetitions, and verify your algorithm's prediction for the LLC size against what the specs of your system say. Write up a report explaining your findings. You can reuse the code and algorithms from the lab experiments.

Deliverables:

1.[Weight: 60%] The source code and execution/compilation instructions/scripts of an application that infers the LLC size of the running system through a timing side-channel algorithm. Use a programming language of your choice that is easy for the markers to test.

2. [Weight: 40%] A maximum 5 page report containing:
   a. The software design, algorithms and configuration parameters of your application
   b. The results you measured
   c. A discussion on how well your algorithm worked based on the experimental results

1 LLC is the highest-level cache, so, if your system has L1, L2 and L3, the LLC will be L3
2 See notes on page 2
3 https://igoro.com/archive/gallery-of-processor-cache-effects/

Non-functional Requirements:

Your application must be written and tested in a Linux environment

Notes

1. Apple M1: For users of Apple's M1 line of CPUs, different modes have different cache line lengths. It has been reported that the M1 has a native 128-byte cache line and that it may use a 64-byte line when running x86 code.
2. Languages: A language closer to the metal like Java, C or C# will work better fothis than an interpreted language like python.

— End of Assignment 4 —

1 LLC is the highest-level cache, so, if your system has L1, L2 and L3, the LLC will be L3
2 See notes on page 2
3 https://igoro.com/archive/gallery-of-processor-cache-effects/