CMP SCI 635: Modern Computer Architecture

Meltdown: Reading Kernel Memory from User Space

Name: Kunjal Panchal Date: 7th Nov, 2019 Student ID: 32126469

Paper: 17 – Security [Lipp17]

Strengths:

1. In order for a Meltdown attack to be worthwhile, either a single byte of data located at a particular memory address or a sequence of consecutive memory addresses (i.e., sequence of bytes) need to be read, so that a meaningful piece of information can be extracted from the data leaked. So if, we can monitor segmentation faults occurring at memory addresses that are close to each other and issue a warning at runtime when these faults become "suspicious", we might be able to prevent the attacks.

2. Meltdown can likely not be exploited from a browser without first also finding a bug in the Javascript engine.

Weaknesses:

1. If the Point of Concept is able to scan thousands of bits/s, how does an attacker make sense of the data E.g. assume I read 12 bytes at (random address X to X+12) and it was 'ABCDABCDABCD' (this could be any arbitrary byte array). As an attacker, how would I make sense of the data? Look for patterns?

2. An OS without the KPTI patch will have the user mode and kernel mode mapped in the same CR3. That CR3 is unique to each process. The kernel will not actually hold sensitive process data inside the kernel image. For example, while the Windows process *lsass.exe* stores your log in info, the kernel is not aware of that and doesn't know where to look for those inside the *lsass* VA space.

Questions/Assertions:

1. Are GPUs exploitable by Meltdown or Spectre? And the performance was GPUs affected by the KAISER patches?

2. "*JavaScript does not provide access to the rdtscp instruction, and Chrome intentionally degrades the accuracy of its high-resolution timer to dissuade timing attacks using performance.now() [1]. However, the Web Workers feature of HTML5 makes it simple to create a separate thread that repeatedly decrements a value in a shared memory location [18, 32]. This approach yielded a high-resolution timer that provided sufficient resolution.*"
   Would it be possible to induce timing from I/O events?

3. Spectre and Meltdown are software to processor attacks. But if your servers are on-premise, an attack is unlikely from serving web page?

4. Like existing NX bit, we can flag certain things as secure/ no speculate/ etc. Prevent speculative execution against flagged areas of memory or by flagged processes, or execute but don't return data (cached or not) until the check is passed.

5. In theory we'd want all our data to be immune to this class of attack. But in practice, most data in memory isn't security-critical, and private user data someone might be concerned about (like certain photos or videos) would take a long time to siphon off using these types of attacks.