ISCAS 中国科学院软件研究所 Institute of Software Chinese Academy of Sciences | 智能软件研究中心 Intelligent Software Research Center

Security for V8：Looking into its Spectre Mitigation

智能软件研究中心 邱吉

qiuji@iscas.ac.cn

2021/04/14

Outline

- What’ s Spectre
- W3C’ s mitigation
- V8’ s mitigation

What’s Spectre (and Meltdown)

- They are vulnerabilities in modern computers leak passwords and sensitive data.

Meltdown

Meltdown breaks the most fundamental isolation between user applications and the operating system. This attack allows a program to access the memory, and thus also the secrets, of other programs and the operating system.

If your computer has a vulnerable processor and runs an unpatched operating system, it is not safe to work with sensitive information without the chance of leaking the information. This applies both to personal computers as well as cloud infrastructure. Luckily, there are software patches against Meltdown.

Spectre

Spectre breaks the isolation between different applications. It allows an attacker to trick error-free programs, which follow best practices, into leaking their secrets. In fact, the safety checks of said best practices actually increase the attack surface and may make applications more susceptible to Spectre

Spectre is harder to exploit than Meltdown, but it is also harder to mitigate. However, it is possible to prevent specific known exploits based on Spectre through software patches.

https://meltdownattack.com

Spectre： how it works

preparing a proper clean cache state

→

instruction trigger speculation execution of CPU

(Conditional branches and indirect jumps)

→

the speculation execution parts

will be canceled later

w/o change to register and memory

w change to L1 cache

→

scanning all of the cache lines to get the one that can hit in L1 cache, then the index of the cache line is the side-channel info

https://meltdownattack.com

To show it more clearly

Prepare cache: contiguously read a large array’ s cache line size aligned elements
Prepare a prob_array which can be cover the whole L1 data cache

trained Cond-Branch or Indirect Jmp which will lead to a transient execution sequence

load rd0, &secret_ptr
load rd1, &probe_array[0]+lowestbyte(rd0)
(rd0/1 is transient because this load will be canceled, but L1 cache is modified)

Read the prob_array’ s cache line size aligned elements and measure the read time to get which line is hit. The line number is the lowestbyte of rd0. Secret leaks!

* secret is usually a out-of-bound index of array

https://meltdownattack.com

Two essential factors

- Factor1: Cond-Branch or Indirect Branch prediction which will lead to false speculation execution parts
- Factor2: Timer with enough resolution to measure L1 Cache hit(nanosecond for a Ghz CPU)

hardware resolution: Factor1

- using barrier instructions to avoid speculatively execute load
- degrade performance by 2x~3x
- not acceptable

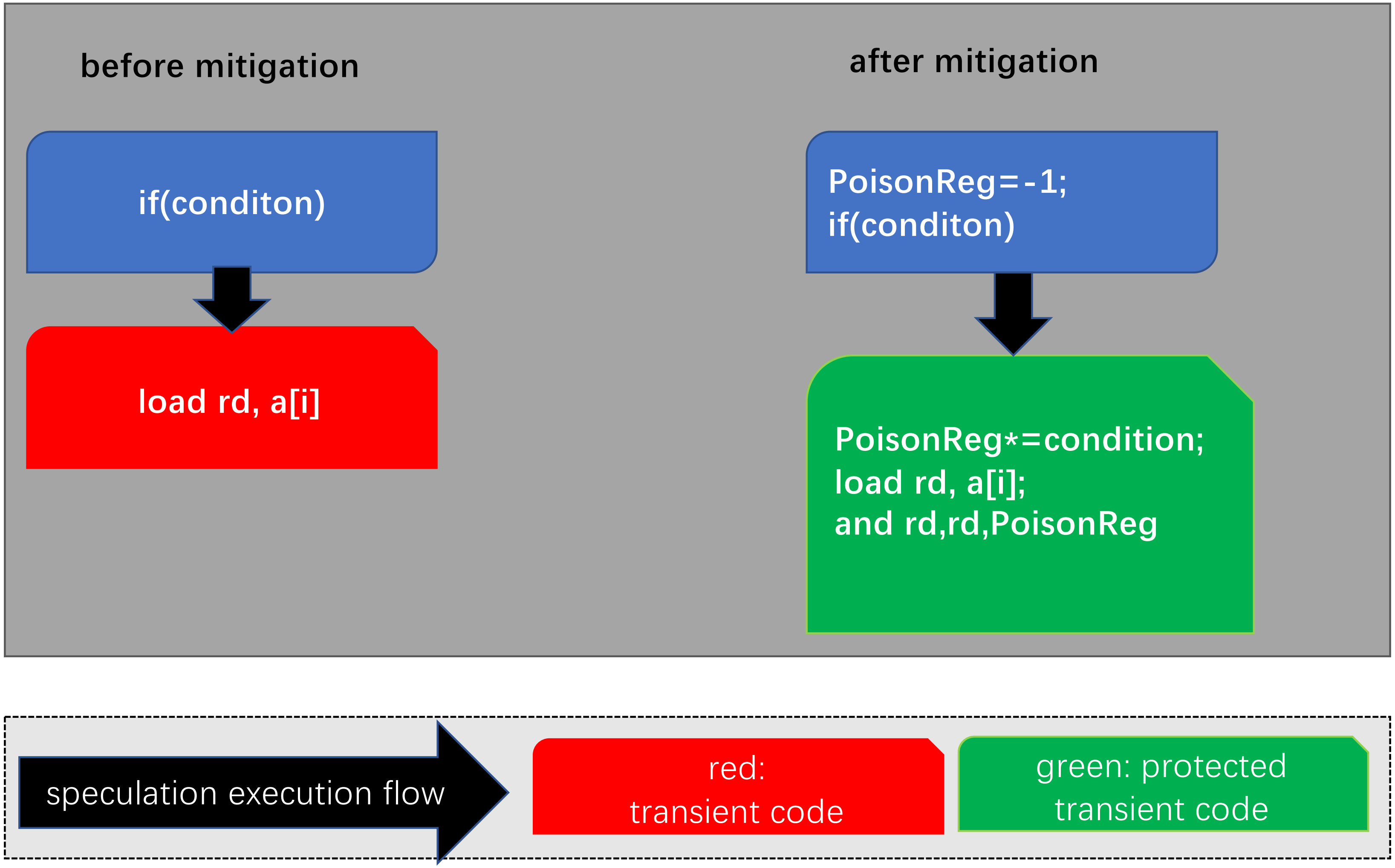
W3C’s mitigation: Factor2

- decrease the resolution of timer API: performance.now() from 5us to 100ms
- disable the SharedArrayBuffer API to prevent construct of high-resolution timer (by measure the clock edge)

V8’s mitigation:

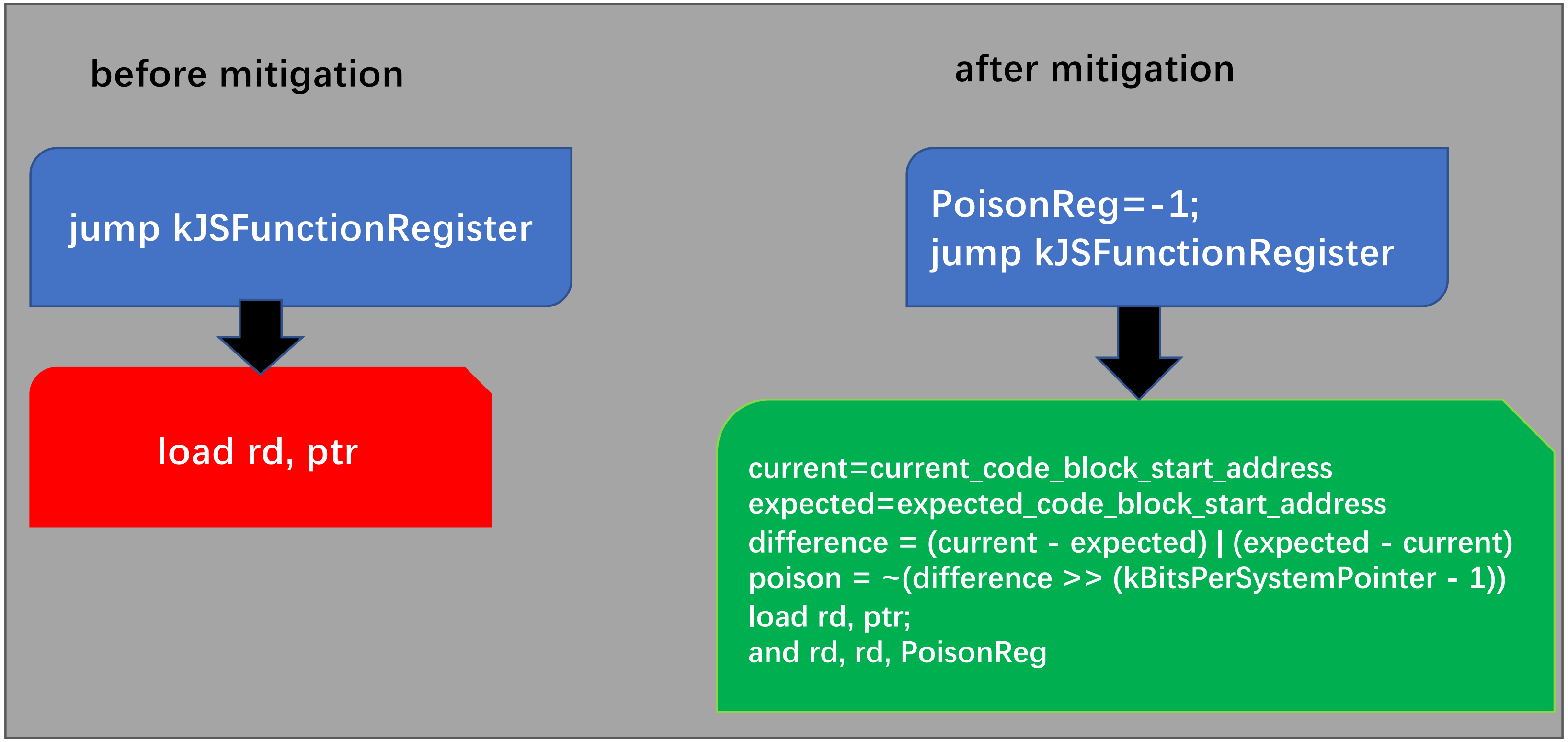
- in JITed code: poison and Retpoline for factor1
- array access bound check
- for 32bit:
  - pad all the array to next power of 2
  - disable the upper bits in user access address (higher half part is kernel address space, for MIPS, 0x8000000-0xffffffff is kernel space)
- for 64bit:
  - by using ALSR

Poison-for cond-branch



If speculation is false, PoisonReg will be 0. Then the load result will be masked out.

Poison-for indirect-jump



If speculation is false, PoisonReg will be 0. Then the load result will be masked out.

Thanks

2020/04/15