Spekulativ udførelse

Er der spøgelser i maskinen?

Kreativ brug af spekulativ udførelse

ind i mellem er der nogen der finder på en virkelig kreativ brug af teknologi. Sådan er det også med spekulativ udførelse. Man kunne jo spørge:

- Instruktioner som bliver fejlforudsagt og derfor bliver annulleret uden at modificere maskinens tilstand, de er jo usynlige. Eller er de?
- Kan man bruge instruktioner som bliver fejforudsagt og annulleret til noget?

De spørgsmål var der flere der stillede sig i løbet af 2017

Svaret åbnede for en hel ny runde af "side-channel attacks"

Paging og page protection (recap)

Vi har en beskyttelsesmekanisme som holder processer adskilt fra hinanden og fra operativsystemet.

- Adresser er virtuelle.
- Hver tilgang til cache indebærer oversættelse fra virtuel til fysisk adresse
- Oversættelse sker via en ATC/TLB (address translation cache/translation lookaside buffer)
- Oversættelse checker lovlighed
- Ulovligt forsøg på adgang sætter en fejl-status på instruktionen

Og særlig for spekulativ udførelse:

- Når en instruktion med fejl-status bliver den ældste og skal fuldføres (Commit)
- Så afbrydes normal udførelse, alle instruktioner smides ud
- Hvorpå processoren skifter til priviligeret "mode" og udfører kode til fejl-håndtering i operativsystemet

Så ulovlig læsning fra lageret kan *ikke* fuldføres og dermed ses udefra. Vel?

Layout af virtuelt addresserum

(liiiige et indskud)

Maskiner har i mange år kunnet skelne mellem "user space" og "kernel space" adresser. Det har gjort det muligt at have begge dele i samme adresserum. For eksempel ku en 32-bit maskine have 2G afsat til kernen og 2G til en kørende proces. Processen kan ikke tilgå kernen fordi kernen er i kernel space og processen er i user-mode.

Når processen laver et systemkald skifter den til kernel-mode og kan tilgå kernel space. Det er praktisk hvis det sker hurtigt - for når man udfører et systemkald vil man gerne tilgå kernens datastrukturer hurtigt. Derfor brugte man at have selve addresse-oversættelsen for kernen sat op, også i user mode.

I mange år blev det anset som uproblematisk - standard practice. Alle vidste jo at oversættelsen fra virtuel til fysisk addresse indeholdt et check af adgangsrettigheder. Og hvis processen forsøgte at læse en adresse i kernel-space, så ville den trigge en page-fault. Og få et gok i nøden!

(Mis)brug af spekulativ udførelse

Betragt flg programstump

```
char meltdown(char* verboten) {
    typedef struct { /* noget der fylder en cacheblok */ } Block;
    Block side_channel[256];
    // kode der med garanti skubber 'side_channel' ud af cachen
    // kode der træner hopforudsigeren så den forudsiger nedenstående forkert
    if ( (* fejlforudsiges "true", men evaluerer laaaangsomt til "false" *) ) {
        // følgende udføres spekulativt og annulleres derpå
        unsigned char probe = *verboten;  // page fault!!!
        side_channel[probe] = 0;
    }
    // mål på tid det tager at tilgå alle elementer i side_channel
    // hvilket element er nu i cache - det der tog kortest tid at tilgå
    return fastest;
}
```

Kapow! Se https://meltdownattack.com/

The Deep Dive

For den teknisk interesserede: Herb Sutter snakker om C++ memory model.

https://www.youtube.com/watch?v=A8eCGOqgvH4 https://www.youtube.com/watch?v=KeLBd2EJLOU

Spørgsmål og Svar