

# Рањивост спекулативног извршавања

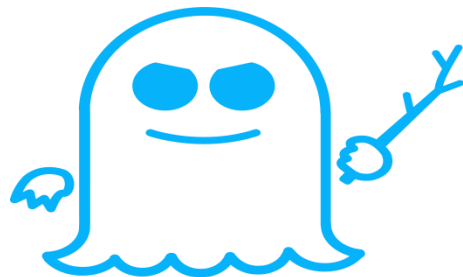


# Садржај

- Цурење информација кроз организацију рачунара
- Рањивост типа *Meltdown*
- Рањивост типа *Spectre*



**MELTDOWN**



**SPECTRE**

# Предвиђање скокова

## • Успешно предвиђање

| инстр. <i>i</i> (branch) | IF | ID | IQ | SC | DI | RR | EX | MEM | WB  | CM  |     |    |
|--------------------------|----|----|----|----|----|----|----|-----|-----|-----|-----|----|
|                          |    | IF | ID | IQ | SC | DI | RR | EX  | MEM | WB  | CM  |    |
|                          |    |    | IF | ID | IQ | SC | DI | RR  | EX  | MEM | WB  | CM |
|                          |    |    |    | IF | ID | IQ | SC | DI  | RR  | EX  | MEM | WB |

| инстр. <i>i</i> (branch) | IF | ID | IQ | SC | DI | RR | EX | MEM | WB  | CM |    |  |
|--------------------------|----|----|----|----|----|----|----|-----|-----|----|----|--|
|                          | IF | ID | IQ | SC | DI | RR | EX | MEM | WB  | CM |    |  |
|                          |    | IF | ID | IQ | SC | DI | RR | EX  | MEM | WB | CM |  |
|                          |    | IF | ID | IO | SC | DI | RR | EX  | MEM | WB | CM |  |

## • Неуспешно предвиђање

Спекулативно извршавање!

| инстр. <i>i</i> (branch) | IF | ID | IQ | SC | DI | RR | EX | MEM | WB   | CM   |      |      |
|--------------------------|----|----|----|----|----|----|----|-----|------|------|------|------|
|                          |    | IF | ID | IQ | SC | DI | RR | EX  | idle | idle | idle |      |
|                          |    |    | IF | ID | IQ | SC | DI | RR  | idle | idle | idle | idle |
|                          |    |    |    | IF | ID | IQ | SC | DI  | idle | idle | idle | idle |

| инстр. <i>i</i> (branch) | IF | ID | IQ | SC | DI | RR | EX | MEM | WB   | CM   |      |  |
|--------------------------|----|----|----|----|----|----|----|-----|------|------|------|--|
|                          | IF | ID | IQ | SC | DI | RR | EX | MEM | idle | idle |      |  |
|                          |    |    | IF | ID | IQ | SC | DI | RR  | idle | idle | idle |  |
|                          |    |    | IF | ID | IQ | SC | DI | RR  | idle | idle | idle |  |

## *Meltdown - Spectre*

- Ради се о два различита типа напада
- Јављају се у више варијанти
- Оба напада омогућавају недозвољени приступ меморији
  - Читање са локације која им не припада
- Оба напада нападају микроархитектуру, а не софтверске грешке
- Оба користе спекулативно извршавање и кеш меморију
  - *Meltdown* напада рад са изузецима
  - *Spectre* напада предвиђање скокова

# Цурење информација

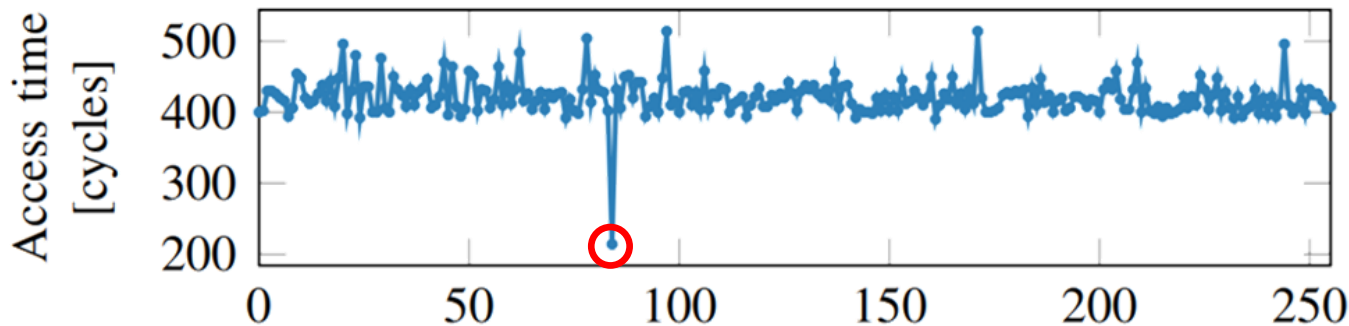
- Шта је напад на бочни (*side*) канала?
  - Прате се функција или карактеристика система који открива нежељене информације
  - Мерење времена за множење ретко поседнуте матрице говори о броју не-нула
  - Време, потрошња енергије, РФ емисије, итд су сви бочни канали
- Шта је прикривени (*covert*) напад на канал ?
  - Мењање система тако да открије информације
  - Инсталирање програма који прати притиске по тастатури
  - *Meltdown* и *Spectre* су прикривени напади на канал
- Ови називи се не користе једнообразно!

# Цурење информација кроз кеш меморију

- Могућност дохватање информација из кеш меморије а да то нико не примети!      Којих информација?
- Да ли се може одредити да ли је адреса кеширана?
- Мерење време прецизним тајмером
  - Некешираној вредности се споро приступа
  - Кешираној вредности се брзо приступа
  - Ово може бити напад и на бочни канал и на прикривени канал

# Напад типа Избаци и учитај – *Flush and Reload*

- Избацили одређени садржај из кеш меморије
- Условно поновно учитавање у кеш
  - Како се условно учитава у кеш?
- Поновни приступ локацији
  - Брз приступ -> адреса је кеширана
  - Спори приступ -> адреса није кеширана



## Рањивост типа *Meltdown*

- Претпоставимо да **злонамерни** код садржи:

```
0 raise_exception
```

```
1 ; rcx = kernel address
```

```
2 ; rbx = probe array
```

```
3 mov al, byte [rcx] <= Зар ово неће генерисати прекид?!
```

```
4 shl rax, 0xc
```

Неће ако не треба да се изврши!

```
5 mov rbx, qword [rbx + rax]
```

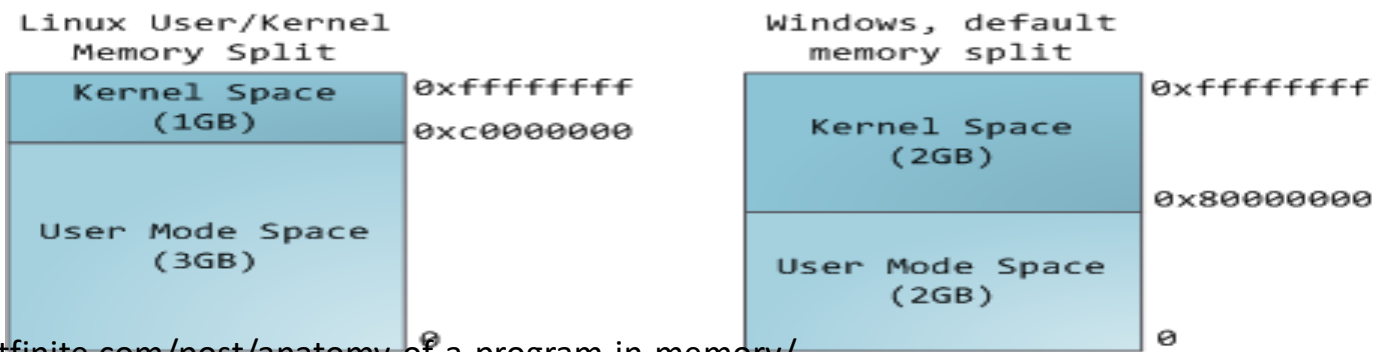


# Рањивост типа *Meltdown*

- Кораци:
- Нападач позива програм
  - Спекулативно извршавање инструкција након позива прекида
    - Чита адресу која припада језгру без провере привилегија
    - Убацује податак у кеш у свој низ
    - Одбацују се промене када се утврди да је ово инструкција прекида
  - Завршава се инструкција позива прекида
- Нападач мери време за приступ до `probe_array[i*4096]`
  - Брзо читање – са локације која одговара податку са локације `kernel_address`
  - Споро читање – са осталих локација
  - Могућност великог броја понављања (  $\sim$  X00KB/s)

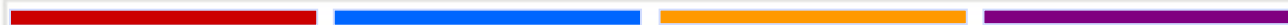
## Рањивост типа *Meltdown*

- Ова рањивост се појављује код Интелових процесора који инструкције извршавају ван редоследа, и код неких Армових процесора, али не и код АМД
- Могуће га је софтверски премостити
  - KPTI/KAISER Linux закрпе које мапирају минимални део језгра са програмом или подацима



# Тровање предвиђања скока

- Могућност да се превари процес тако да почне да одаје информације
- Може се тровати:
  - испуњеност услова скока
  - адреса на коју се скаче



# Шта се дешава приликом промене контекста?

- Оперативни систем је задужен за промену контекста, док је хардвер задужен само за генерисање прекида
- Сваком процесу су придружене структуре података које прате његово релевантно сање:
  - **Стање архитектуре рачунара:** РС, регистри, РТР
    - Чувају се и рестаурирају приликом промене контекста
    - Стање меморије?
  - **Стање организације рачунара:** кеш, предвиђања, ...
    - Игнорише се или се брише
    - Игнорисање је кључни аспект код ових рањивости!

## Рањивост типа *Spectre*

- Претпоставимо да **кориснички** код садржи:  

```
if (x < array1_size)  
    y = array2[array1[x]*4096];
```
- При чему x потиче из злонамерног извора
- Извршавање без предвиђања скока је безбедно
  - Процесор никада неће извршити `array2[array1[x]*4096]` уколико је `x < array1_size`
- Шта ће се десити код спекулативног извршавања?

## Рањивост типа *Spectre*

- Претпоставимо да **кориснички** код садржи:  

```
if (x < array1_size)  
    y = array2[array1[x]*4096];
```
- Пре напада:
  - Утренирати предвиђање скока да очекује да услов буде испуњен
  - Избацити `array1_size` и `array2[]` из кеша

# Рањивост типа *Spectre*

- Претпоставимо да **кориснички** код садржи:  

```
if (x < array1_size)  
    y = array2[array1[x]*4096];
```
- Корази:
- Нападач позива програм при чему поставља да је  $x=N$  ( $N > \text{array1\_size}$ )
  - Спекулативно извршавање чека на `array1_size`
    - Предвиђа да ће услов *if* бити испуњен
    - Чита адресу (`array1 + x`) без провера границе за  $x$
    - Читање податка који је у кешу
    - Чита меморију са адресе (`array2 base + N*4096`)
    - Убацује `array2[N*4096]` у кеш
    - Рачуна да услов *if* није испуњен и одбацује промене
  - Завршава се инструкција
- Нападач мери време за приступ до `array2[i*4096]`
  - Читање са  $N$  је брзо остало је споро
  - Могућност великог броја понављања (  $\sim 10\text{KB/s}$  )

## Рањивост типа *Spectre 2*

- Коришћење инструкције срачунатог скока (`jmp [rax]`)
- Затровати предвиђање скока тако да се скочи на код који одаје садржај меморије (претходни пример)!
- Кораци:
  - Затровати адресу на коју се скаче код предвиђање скока тако да се скочи на део кода који одаје садржај меморије
  - Испразнити кеш
  - Покренути код
  - Читати осетљиве податке
  - Понављати у петљи



## Рањивост типа *Spectre 2'*

- Злонамерни код:

Label0: if (1)

Label1: ...

- Кориснички код:

R1  $\leftarrow$  (from attacker)

R2  $\leftarrow$  some secret

Label0: if (...) {...}

else {...}

...

Label1: lw [R2]

- Затровати предвиђање скока тако да се скочи на код који одаје садржај меморије (претходни пример)!

## Рањивост типа *Spectre*

- Како затровати *BHT/BTB*?
  - Није тешко ако су позати алгоритми и ако се пише код који то ради
    - Могуће и коришћењем *JavaScript*, за *VM* на *EC2*
- Како натерати рачунар који се напада да покрене овакав код?
  - Зависи од тога ко је жртва
  - Најлакше у окружењу за *JIT*, могуће је читати стање *Chrome* користећи *JavaScript*

## Рањивост типа *Spectre*

- Рањивост типа *Spectre* погађа све процесоре са извршавањем ван редоследа!
  - Потврђено код Интела, Арма, Аемдеа
- Софтверске закрпе нису довољне и споре су
  - Смањење индиректних скокова
  - Забранити спекулативно извршавање код критичних делова кода
- Није лако одбранити се од ових напада!



# Ко је откио рањивост?

- *Meltdown*
  - Независно су открили:
    - John Horn (Google Project Zero)
    - Werner Haas, Thomas Prescher (Cyberus Technology)
    - Daniel Gruss, Moritz Lipp, Stefan Mangard, Michael Schwarz (Graz University of Technology)
- *Spectre*
  - Независно су га открили:
    - John Horn (Google Project Zero)
    - Paul Kocher у сарадњи са: Daniel Genkin (University of Pennsylvania and University of Maryland), Mike Hamburg (Rambus), Moritz Lipp (Graz University of Technology), Yuval Yarom (University of Adelaide and Data61)

# Питања?

Електротехнички Факултет  
Универзитет у Београду

