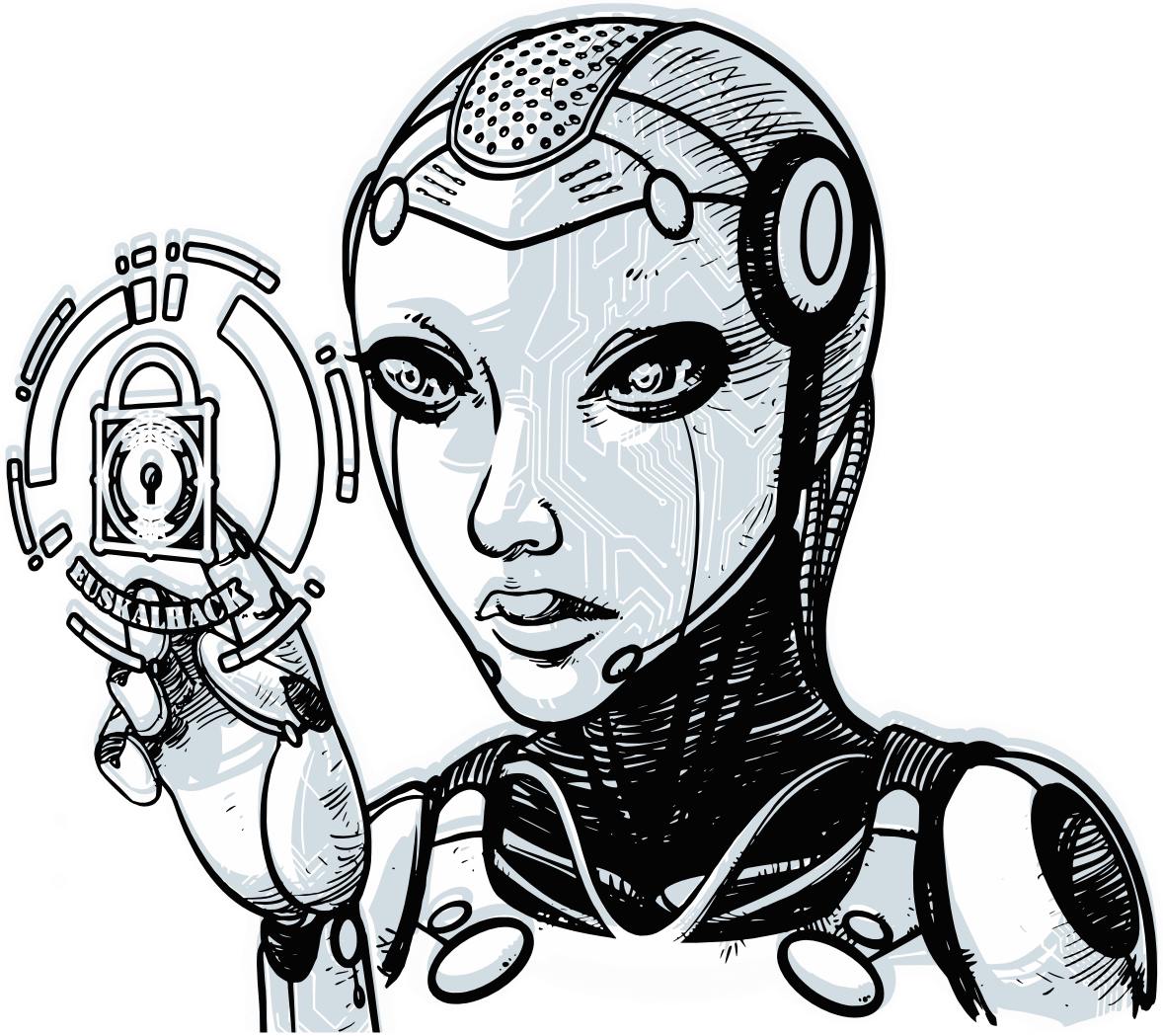
EuskalHack Security Congress VII



$ whoami

Name: Pepe Vila
Alias: cgvwzq
Web: https://vwzq.net/

In a past life: XSS troll master, NFC tinkerer, web security, CTFs, 2-columns-paper writer…

These days: security architect at Arm ¯\(°_o)/¯

DISCLAIMER: this presentation reflects my personal views

Most of which are nothing but a mix of other people's views incorporated into my neural network as result of a random set of experiences…

In a hole in the ~~micro~~architecture there lived a vulnerability...

An architect works on the architecture

~~as an engineer works on the …~~

The Architecture <u>defines the behavior of an abstract machine:</u>

- **ISA, memory model, execution modes, I/O, etc.**

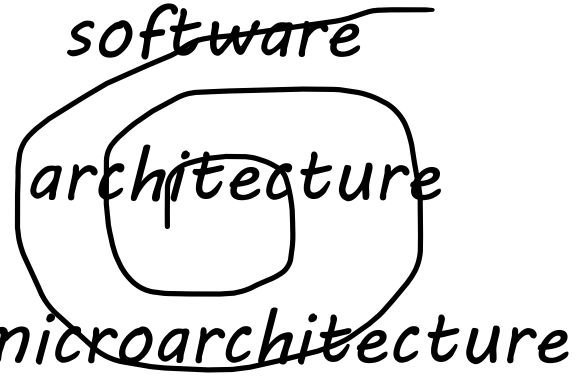
Microarchitecture refers to the implementation details:

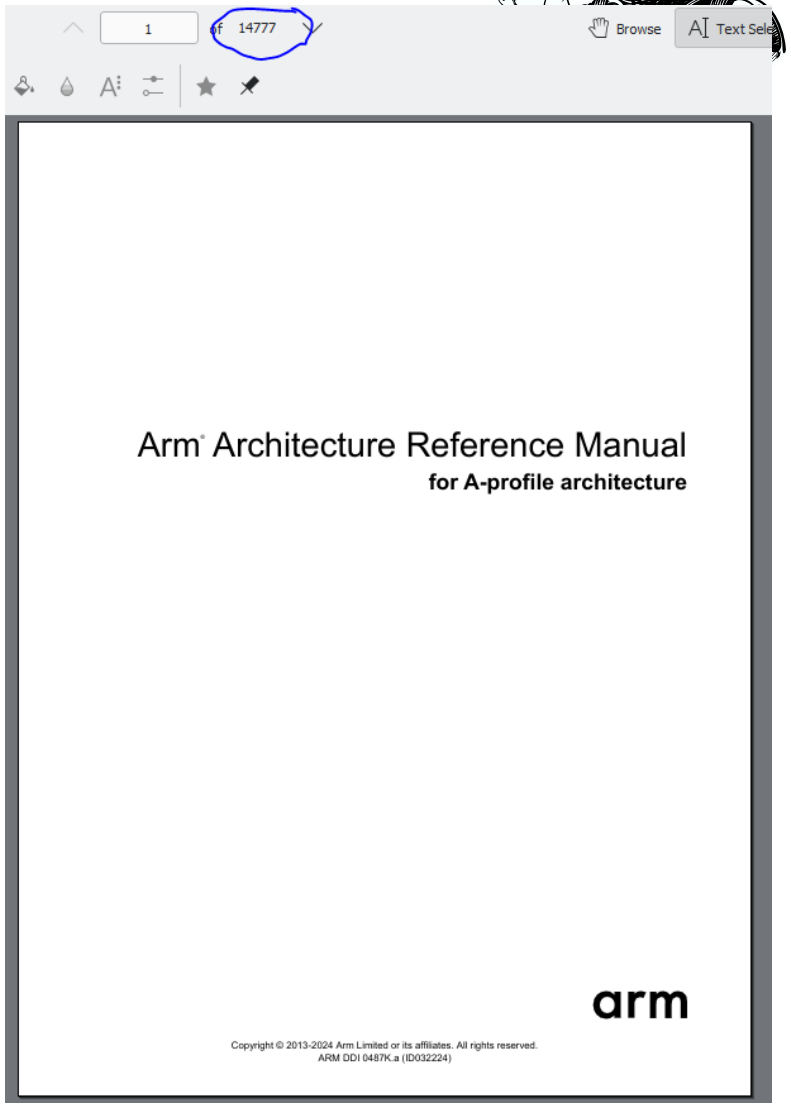
- **caches, execution pipelines, optimizations, etc.**

software

architecture

microarchitecture



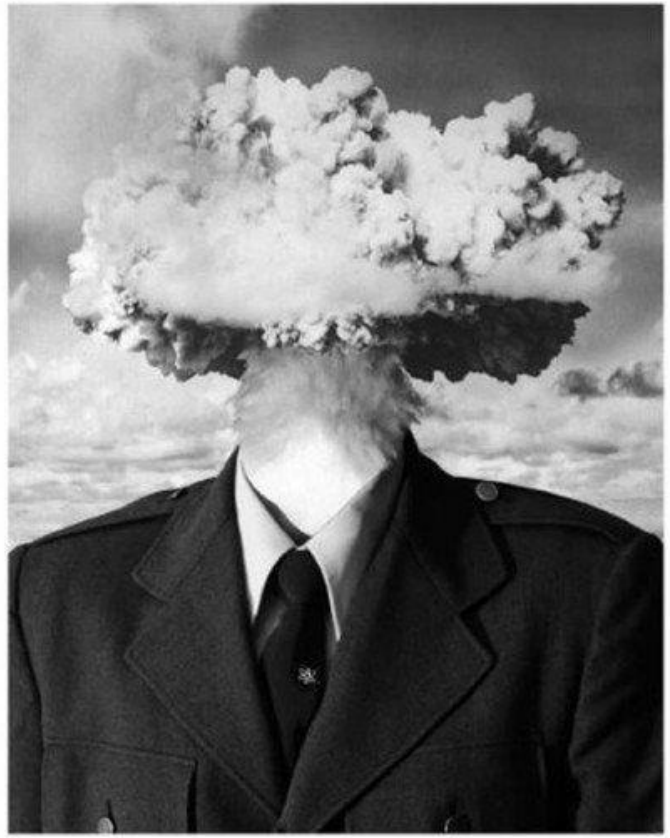


https://developer.arm.com/documentation/ddi0487/latest/

CPU vulns:There and back again

And security?

2017: Spectre v1 - Bound check bypass
2017: Spectre v2 - Branch target
2017: Meltdown - Rogue data cache load
2018: Rogue system register read
2018: Spectre v4 - Speculative Store bypass
2018: Lazy FP state restore
2018: Spectre v1.1 - bound check bypass store
2018: SpectreRSB/ret2spec - return mispredict
2018: Foreshadow - L1 terminal fault
2019: MDS: RIDL, Zombieload, Fallout, CacheOut, …
2020: Load Value Injection
2022: MMIO stale data issues
2022: PACMAN
2022: Branch type confusion: retbleed, phantom jumps..
2022: Downfall
2022: Augury
2022: Spectre BHB
2023: Zenbleed
2023: Inception



Some (assumed) background

- Pipelining, out-of-order execution, speculation (e.g., branch prediction)

One useful lie

For simplicity we'll replace the exfiltration channels by a simple **oracle**

- we can ignore whether we do Flush+Reload, Prime+Probe, TLB covert-channel, etc.

Today, the attacker can **magically observe the addresses of all locations accessed** by the victim, but not the values.

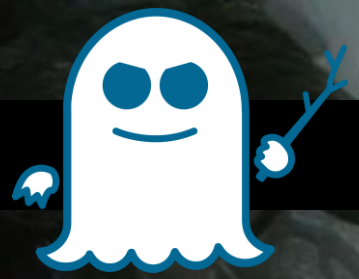
Remember: Local attacker with code execution.

"Priestess of Delphi" by John Collier, 1891.

CPU vulns:There and back again

Spectre Attacks: Exploiting Speculative Execution

Paul Kocher[1], Jann Horn[2], Anders Fogh[3], Daniel Genkin[4],
Daniel Gruss[5], Werner Haas[6], Mike Hamburg[7], Moritz Lipp[5],
Stefan Mangard[5], Thomas Prescher[6], Michael Schwarz[5], Yuval Yarom[8]
[1] Independent (www.paulkocher.com), [2] Google Project Zero,
[3] G DATA Advanced Analytics, [4] University of Pennsylvania and University of Maryland,
[5] Graz University of Technology, [6] Cyberus Technology,
[7] Rambus, Cryptography Research Division, [8] University of Adelaide and Data61



Spectre v2

Spectre v2 (aka. Branch target injection)

Although more generally, we should say prediction injection

Attacker

```
0x500: ret
...
0x700: mov x0, #0x500
0x704: blr x0
0x708: blr x0
0x70c: blr x0
...
```

Branch predictor

PC=0x704	Target=0x500
PC=0x708	Target=0x500
PC=0x70c	Target=0x500

Victim

```
; exfiltration gadget
0x500: ldr x0, [x0]
0x504: ldr x1, [x0]
...

; attacker controls x0
0x704: ldr x4, [x3]
0x708: blr x4
```

Arbitrary read

Exfiltration

Long latency operation…

Prediction hit

Spectre v2

Spectre v2 (aka. Branch target injection)

Although more generally, we should say prediction injection

Attacker

```
0x500: ret
...
0x700: mov x0, #0x500
0x704: blr x0
0x708: blr x0
0x70c: blr x0
...
```

Value coming from memory to resolve pending load (dramatization)

Branch predictor

PC=0x704	Target=0x500
PC=0x708	Target=0x500
PC=0x70c	Target=0x500

```
; attacker controls x0
0x704: ldr x4, [x3]
0x708: blr x4
...
```

Long latency operation…

Spectre v2 (aka. Branch target injection)

Although more generally, we should say prediction injection

Branch predictor

PC=0x704	Target=0x500
PC=0x708	Target=0x500
PC=0x70c	Target=0x500

Victim

```
; exfiltration gadget
0x500: ldr x0, [x0]
0x504: ldr x1, [x0]
...

; attacker controls x0
0x704: ldr x4, [x3] ; x4=0x200
0x708: blr x4
```

Attacker

```
0x500: ret
...
0x700: mov x0, #0x500
0x704: blr x0
0x708: blr x0
0x70c: blr x0
...
```

Prediction was wrong, flush

Whose problem is it?

B2.9.4 “Restrictions on the effects of speculation from Armv8.5”

If FEAT_CSV2 is implemented:

- Code running in one hardware-defined context (context1) cannot either exploitatively control, or predictively leak to, the speculative execution of code in a different hardware-defined context (context2), as a result of the behavior of any of the following resources:
  - Branch target prediction based on the branch targets used in context1.
    - This applies to both direct and indirect branches, including return instructions, but excludes the prediction of the direction of a conditional branch.
  - Data Value predictions based on data value from execution in context1.

    Note

    `PSTATE.{N,Z,C,V}` values from context1 are not considered a data value for this purpose.

  - Virtual address-based cache prefetch predictions generated as a result of execution in context1, based on, or causing dereference of, data values from memory.
  - Any other prediction mechanisms, other than Branch, Data Value, or Cache Prefetch predictions.

  In this definition, the hardware-defined context is determined by:

  - The Exception level.
  - The Security state.
  - When executing at EL1, if EL2 is implemented and enabled in the current Security state, the VMID.
  - When executing at EL0, whether the EL1&0 or the EL2&0 translation regime is in use.
  - When executing at EL0 and using the EL1&0 translation regime, the *address space identifier* (*ASID*) and, if EL2 is implemented and enabled in the current Security state, the VMID.
  - When executing at EL0 and using the EL2&0 translation regime, the ASID.

In practice...

An example of implementation that meets the requirements is **context tagging**

Branch predictor

Tag=A	PC=0x700	Target=0x500
Tag=A	PC=0x704	Target=0x500
Tag=A	PC=0x708	Target=0x500

Victim

```
; exfiltration gadget
0x500: ldr x0, [x0]
0x504: ldr x1, [x0]
...

; attacker controls x0
0x704: ldr x4, [x3]
0x708: blr x4
...
```

Prediction miss

Attacker

```
0x500: ret
...
0x700: mov x0, #0x500
0x704: blr x0
0x708: blr x0
0x70c: blr x0
...
```

CPU vulns:There and back again

Branch History Injection: On the Effectiveness of Hardware Mitigations Against Cross-Privilege Spectre-v2 Attacks

Enrico Barberis† Pietro Frigo† Marius Muench Herbert Bos Cristiano Giuffrida

Vrije Universiteit Amsterdam
{e.barberis, p.frigo, m.muench}@vu.nl
{herbertb, giuffrida}@cs.vu.nl

Spectre BHB

Spectre BHB (aka. Branch History Injection)

or branch target or (prediction) reuse attacks

Branch history

…	0	1	0	0	1	1	0	1

PC

h

Branch predictor

Tag=V	Target=0x100
Tag=A	Target=0x500
Tag=V	Target=0x120
Tag=V	Target=0x300
Tag=V	Target=0x340

Victim

```
; function with gadget
0x340:
...
0x384: ldr x1, [x7]
0x388: ldr x1, [x1]
...

; attacker controls x7
0x704: ldr x4, [x3]
0x708: blr x4
...
```

Attacker

```
; modify branch history
ldrb w3, [x0]
add x0, x0, #1
tbnz w3, #0, +4
ldrb w3, [x0]
add x0, x0, #1
tbnz w3, #0, +4
...
```

Arbitrary read

Exfiltration

Prediction hit

Whose problem is it?

If either FEAT_CSV2_1p1 or FEAT_CSV2_3 is implemented, code running in one hardware-defined context (context1) cannot either exploitatively control, or predictively leak to, the speculative execution of code in a different hardware-defined context (context2) as a result of the behavior of branch target prediction based on the branch history used in context1.

Branch prediction

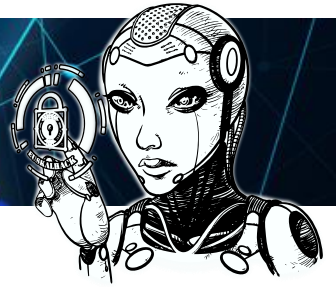
If FEAT_CLRBHB is not implemented, then the architecture does not define any branch predictor maintenance instructions for AArch64 state.

If branch prediction is architecturally visible, cache maintenance must also apply to branch prediction.

When FEAT_CLRBHB is implemented, the CLRBHB instruction available. When the CLRBHB instruction is executed, the branch history is cleared for the current context to the extent that branch history information created before the CLRBHB instruction cannot be used by code before the CLRBHB instruction to exploitatively control the execution of any code in the current context appearing in program order after the instruction.

When FEAT_ECBHB is implemented, the branch history information created in a context before an exception to a higher Exception level using AArch64 cannot be used by code before that exception to exploitatively control the execution of any indirect branches in code in a different context after the exception.

CPU vulns:There and back again



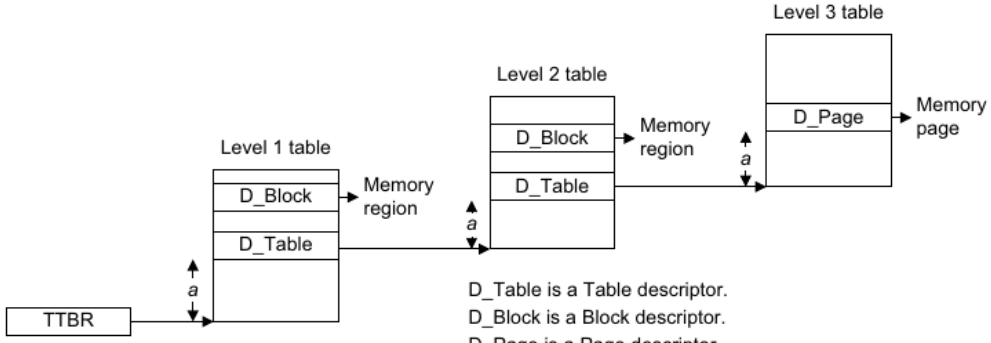
Meltdown: Reading Kernel Memory from User Space

Moritz Lipp[1], Michael Schwarz[1], Daniel Gruss[1], Thomas Prescher[2],
Werner Haas[2], Anders Fogh[3], Jann Horn[4], Stefan Mangard[1],
Paul Kocher[5], Daniel Genkin[6,9], Yuval Yarom[7], Mike Hamburg[8]
[1]*Graz University of Technology,* [2]*Cyberus Technology GmbH,*
[3]*G-Data Advanced Analytics,* [4]*Google Project Zero,*
[5]*Independent (www.paulkocher.com),* [6]*University of Michigan,*
[7]*University of Adelaide & Data61,* [8]*Rambus, Cryptography Research Division*

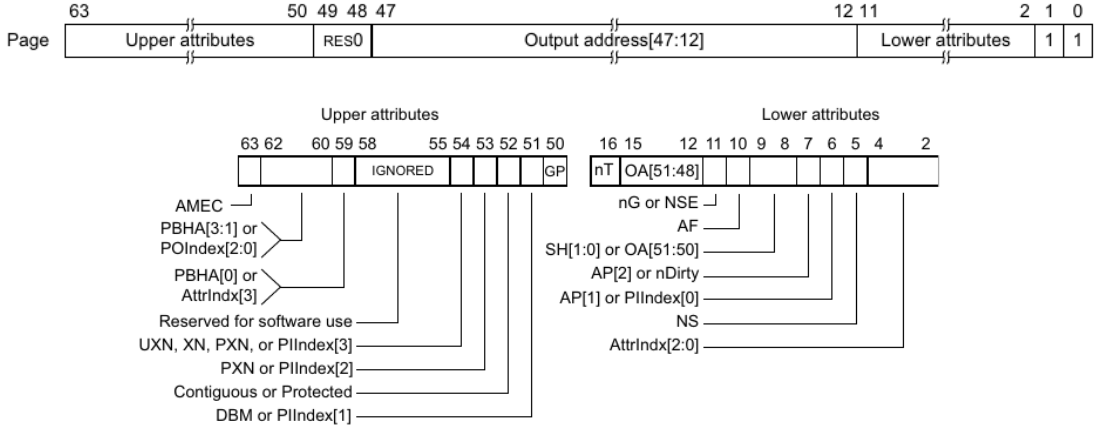
Meltdown

Meltdown

```
ldrb w0, [x1]       // x1 = 0xffff000000000abcd
ldr xzr, [x2, x0]
```



4KB granule 48-bit OA



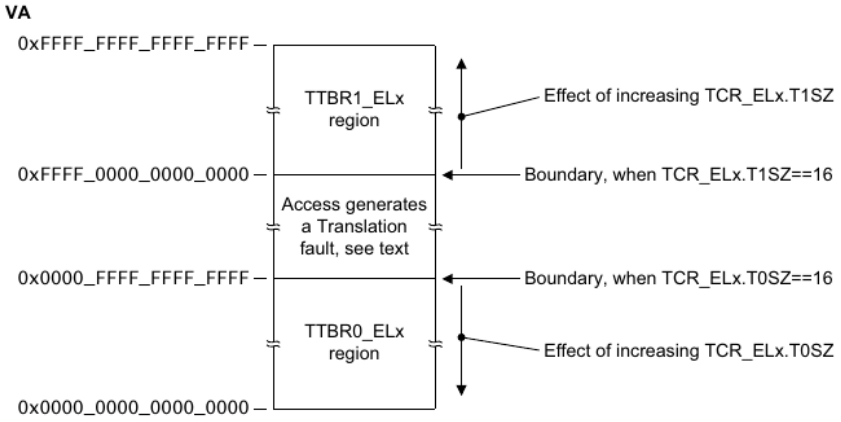


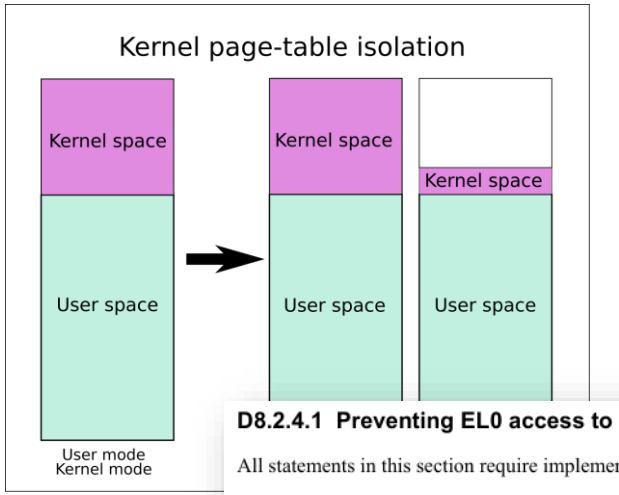
Table D8-62 Summary of possible memory access permissions using Direct permissions for a stage 1 translation supporting two Exception levels

UXN	PXN	AP[2:1]	WXN	Permission
0	0	00	0	PrivRead, PrivWrite, PrivExecute, UnprivExecute
0	0	00	1	PrivRead, PrivWrite, PrivWXN, UnprivExecute
0	0	01	0	PrivRead, PrivWrite, UnprivRead, UnprivWrite, UnprivExecute
0	0	01	1	PrivRead, PrivWrite, UnprivRead, UnprivWrite, UnprivWXN
0	0	10	x	PrivRead, PrivExecute, UnprivExecute
0	0	11	x	PrivRead, PrivExecute, UnprivRead, UnprivExecute
0	1	00	x	PrivRead, PrivWrite, UnprivExecute
0	1	01	0	PrivRead, PrivWrite, UnprivRead, UnprivWrite, UnprivExecute
0	1	01	1	PrivRead, PrivWrite, UnprivRead, UnprivWrite, UnprivWXN
0	1	10	x	PrivRead, UnprivExecute
0	1	11	x	PrivRead, UnprivRead, UnprivExecute
1	0	00	0	PrivRead, PrivWrite, PrivExecute
1	0	00	1	PrivRead, PrivWrite, PrivWXN
1	0	01	x	PrivRead, PrivWrite, UnprivRead, UnprivWrite
1	0	10	x	PrivRead, PrivExecute
1	0	11	x	PrivRead, PrivExecute, UnprivRead
1	1	00	x	PrivRead, PrivWrite
1	1	01	x	PrivRead, PrivWrite, UnprivRead, UnprivWrite
1	1	10	x	PrivRead
1	1	11	x	PrivRead, UnprivRead

If a stage 1 translation regime supports two VA ranges, then all of the following are used to select the TTBR_ELx

- If VA bit[55] is zero, then TTBR0_ELx is selected.
- If VA bit[55] is one, then TTBR1_ELx is selected.

Whose problem is it?



https://en.wikipedia.o

D8.2.4.1 Preventing EL0 access to halves of the address map

All statements in this section require implementation of FEAT_E0PD.

The TCR_ELx.{E0PD0, E0PD1} fields can be used to prevent EL0 access to the addresses translated by the corresponding TTBR0_ELx or TTBR1_ELx.

When the TCR_ELx.{E0PD0, E0PD1} fields prevent EL0 access to an address translated by TTBR0_ELx or TTBR1_ELx, then a level 0 Translation fault is generated.

When the TCR_ELx.{E0PD0, E0PD1} fields generate a level 0 Translation fault, then all of the following apply:

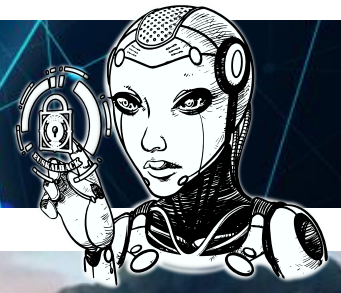
- The time needed to take the fault should be the same whether or not the address accessed is present in a TLB, to mitigate attacks that use fault timing.
- The fault generated does not affect any micro-architectural state of the PE in a manner that is different if the address accessed is present in a TLB or not, to prevent this information being used to determine the presence of the address in a TLB.

When the TCR_ELx.{E0PD0, E0PD1} fields generate a level 0 Translation fault, the fault is not counted as a TLB miss for performance monitoring features.

B2.9.4 "Restrictions on the effects of speculation from Armv8.5"

FEAT_CSV3 introduces these restrictions:

- Data loaded under speculation with a Permission or Domain fault cannot be used to form an address, generate condition codes, or generate SVE predicate values to be used by other instructions in the speculative sequence.
- Any read under speculation from a register that is not architecturally accessible from the current Exception level cannot be used to form an address, to generate condition codes, or to generate SVE predicate values to be used by other instructions in the speculative sequence.



Augury: Using Data Memory-Dependent Prefetchers to Leak Data at Rest

Jose Rodrigo Sanchez Vicarte[1*], Michael Flanders[1†], Riccardo Paccagnella*, Grant Garrett-Grossman*, Adam Morrison[‡], Christopher W. Fletcher*, David Kohlbrenner[†]
*University of Illinois Urbana-Champaign, [‡]Tel Aviv University, [†]University of Washington
{josers2, rp8, grantlg2, cwfletch}@illinois.edu, mad@cs.tau.ac.il, {mkf727, dkohlbre}@cs.washington.edu

GoFetch: Breaking Constant-Time Cryptographic Implementations Using Data Memory-Dependent Prefetchers

Boru Chen *UIUC* · Yingchen Wang *UT Austin* · Pradyumna Shome *Georgia Tech* · Christopher W. Fletcher *UC Berkeley*
David Kohlbrenner *University of Washington* · Riccardo Paccagnella *Carnegie Mellon University* · Daniel Genkin *Georgia Tech*

Data prefetchers

Data prefetchers

Stride prefetcher


```
for (int i=0; i<size; i+=stride) {
   sum += array[i];
}
```

Replay (or SMS) prefetcher


```
int process(my_struct *s) {
   s->out = s->field1 + s->field2;
}
```

Adjacent prefetcher


```
for (int i=0; i<size; i++) {
   sum += array[i];
}
```

Pointer (or DMP) prefetcher


```
for (int i=0; i<2; i++) {
   sum += arr[i]->val;
}
```

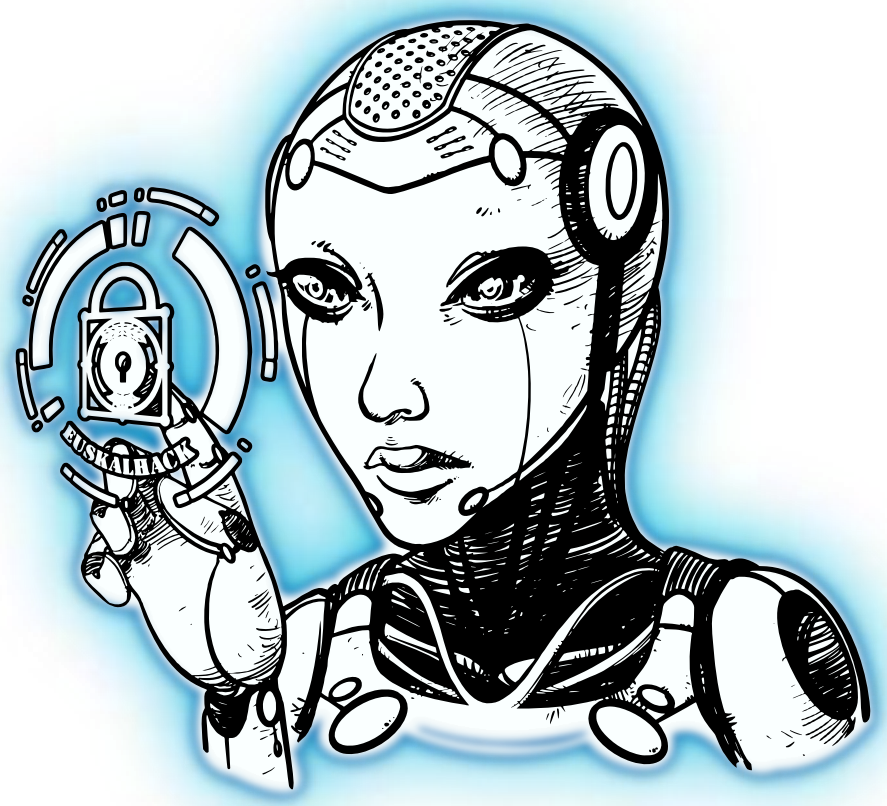
Concluding thoughts…

- Hardware security issues are not more complex than software
  - It's just a blacker blackbox
  - Different background, but offensive mindset translates well

- Researchers (and attackers?) understanding of microarchitectures has increased dramatically → We are seen increasingly complex exploits
  - Keep breaking assumptions about what is and what is not exploitable

- Unexplored surface attacks → Lots of opportunities
  - More and more accelerators (GPUs, NPUs, JPegUs? Dafuq! :S)
  - New vendors building their custom silicon

- Increasing number of hardware security features: PAN, PAC, BTI, MTE, PPL, APRR, KTTR..
  - Most bypasses exploit deployment/configuration issues
  - PAN bypass was an interesting case of an architectural (i.e., design) problem
  - Will we start seeing hardware vulns in exploit chains?



¡MUCHAS GRACIAS!

ESKERRIK ASKO!