A little less SPECULATION, a little more ACTION (please)

mitigating speculative execution side-channel vulnerabilities in Fuchsia

Matthew Riley
mattdr@google.com
@mdriley25519

opening credits

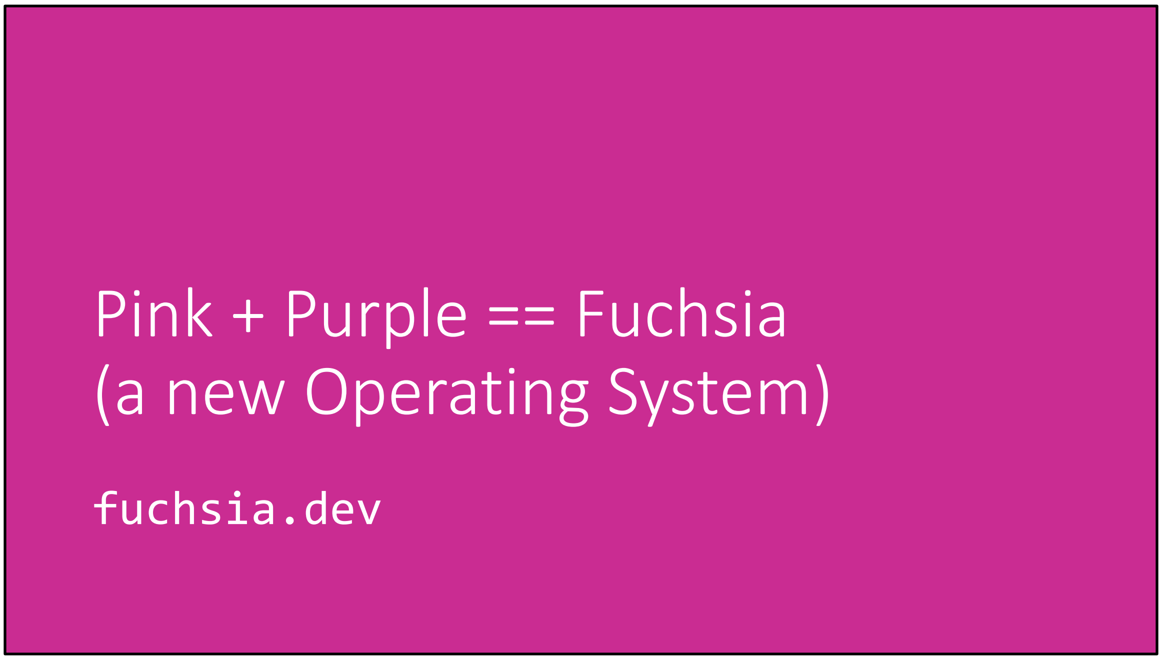
Venkatesh Srinivas

Anthony Steinhauser

read the fine manuals

Intel 64 and IA-32 Architectures Optimization Reference Manual
Intel 64 and IA-32 Architectures Software Developer's Manual
Intel 64 and IA-32 Architectures Software Developer's Manual
Intel 64 and IA-32 Architectures Software Developer's Manual
Intel 64 and IA-32 Architectures Software Developer's Manual
Intel 64 and IA-32 Architectures Software Developer's Manual

obligatory collage of ATTACK LOGOS

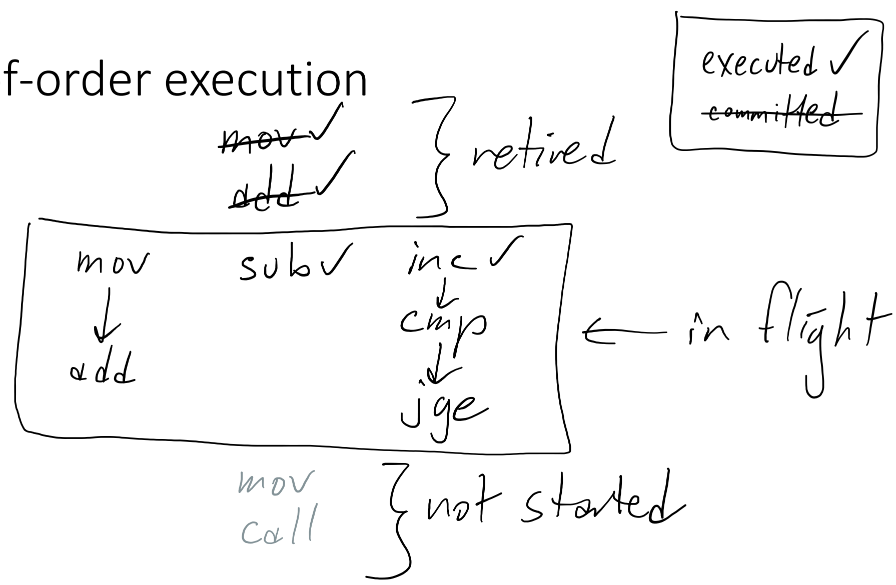


Pink + Purple == Fuchsia (a new Operating System)

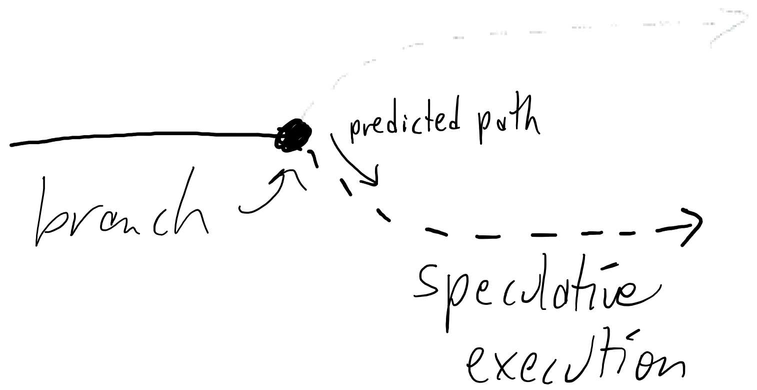
`fuchsia.dev`

https://fuchsia.dev
https://fuchsia.googlesource.com

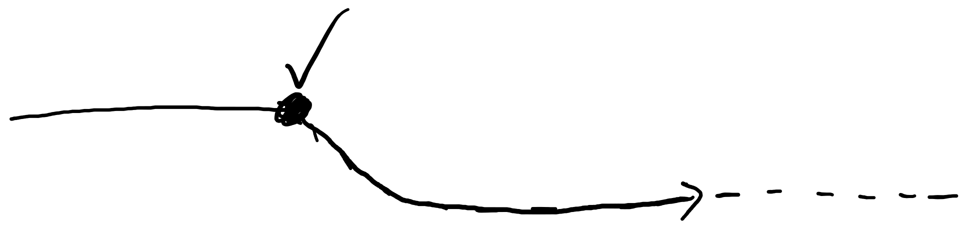
out-of-order execution



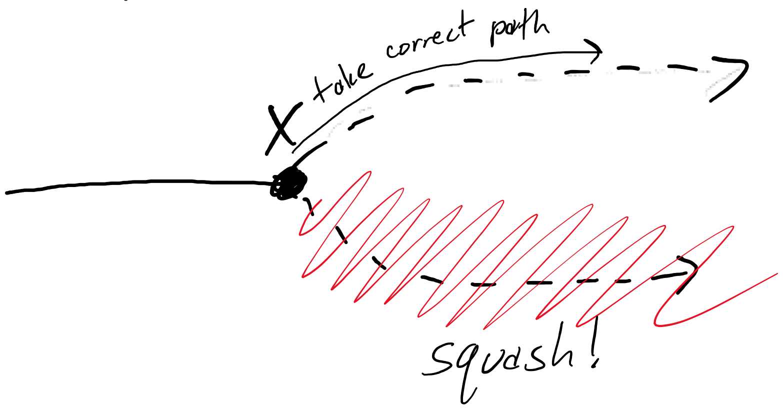
branch prediction



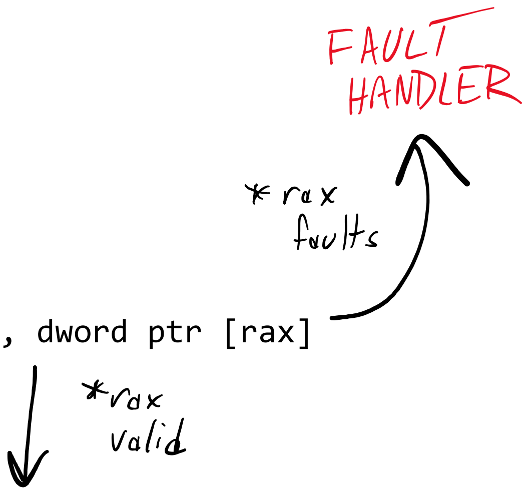
branch prediction – success



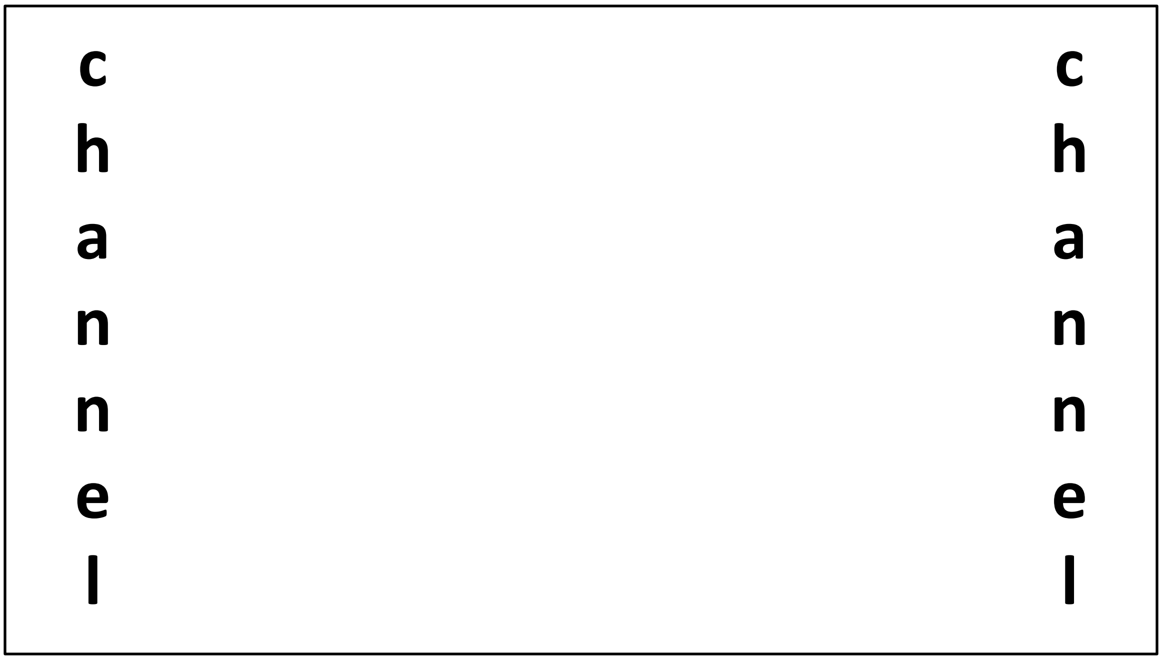
branch prediction – oh no



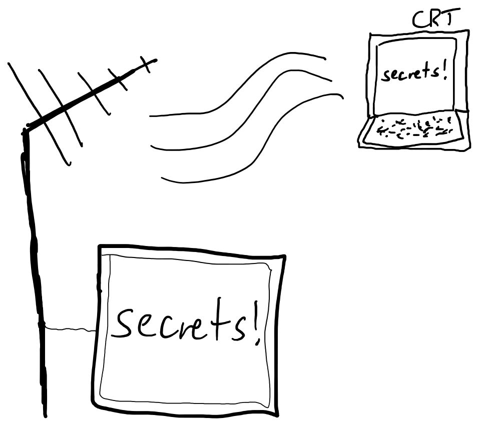
optimism



```
mov      ecx, dword ptr [rax]
```



Sorry.



https://en.wikipedia.org/wiki/Van_Eck_phreaking

CACHE MISSING FOR FUN AND PROFIT

COLIN PERCIVAL

ABSTRACT. Simultaneous multithreading — put simply, the shar-
ing of the execution resource
multiple execution threads —
its introduction (under the
Pentium 4 processors. In th
ficiency and economy of pro
resources between threads ex
particular concern is that the
caches.
We demonstrate that this
vides not only an easily used
tween threads, but also perm
theory, with limited privileges
thread, allowing in many cas
Finally, we provide some s
erating system vendors, and t
of how this attack could be m

1. INT

As integrated circuit fabricati
ing not only faster transistors bu
ers have been met with two criti
have increased dramatically in
easy to spend extra transistors

Cache-timing attacks on AES

Daniel J. Bernstein *

Department of Mathematics, Statistics, and Computer Science (M/C 249)
The University of Illinois at Chicago
Chicago, IL 60607–7045
djb@cr.yp.to

Abstract. This paper demonstrates complete AES key recovery from known-plaintext timings of a network server on another computer. This attack should be blamed on the AES design, not on the particular AES library used by the server; it is extremely difficult to write constant-time high-speed AES software for common general-purpose computers. This paper discusses several of the obstacles in detail.

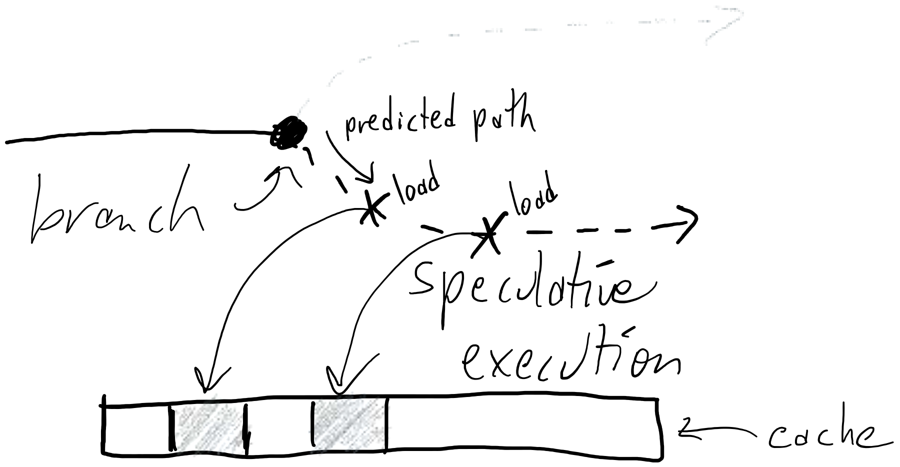
Keywords: side channels, timing attacks, software timing attacks, cache timing, load timing, array lookups, S-boxes, AES

1 Introduction

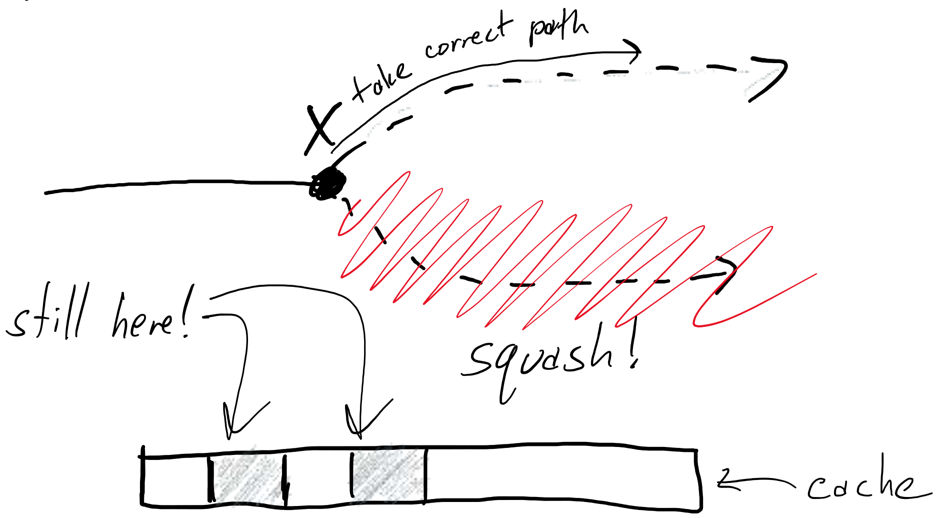
This paper reports successful extraction of a complete AES key from a network

https://www.daemonology.net/papers/htt.pdf
https://cr.yp.to/antiforgery/cachetiming-20050414.pdf

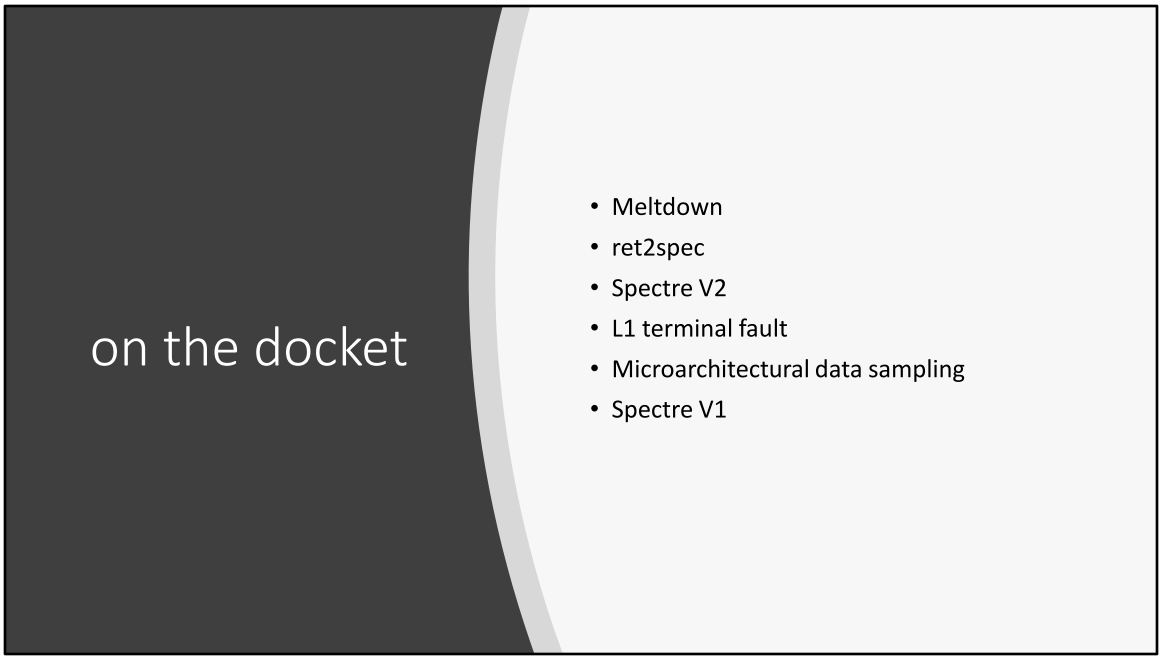
speculative leak via cache side-channel



speculative leak via cache side-channel

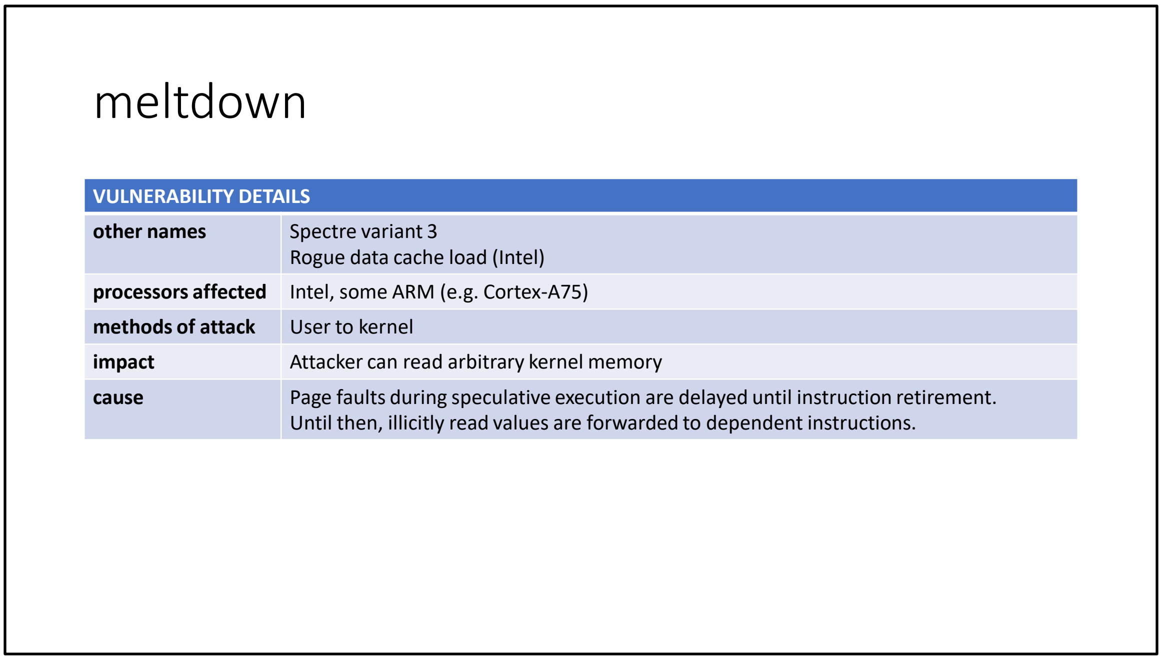


https://googleprojectzero.blogspot.com/2018/01/reading-privileged-memory-with-side.html
https://spectreattack.com/spectre.pdf



on the docket

- Meltdown
- ret2spec
- Spectre V2
- L1 terminal fault
- Microarchitectural data sampling
- Spectre V1



meltdown

VULNERABILITY DETAILS	
other names	Spectre variant 3 Rogue data cache load (Intel)
processors affected	Intel, some ARM (e.g. Cortex-A75)
methods of attack	User to kernel
impact	Attacker can read arbitrary kernel memory
cause	Page faults during speculative execution are delayed until instruction retirement. Until then, illicitly read values are forwarded to dependent instructions.

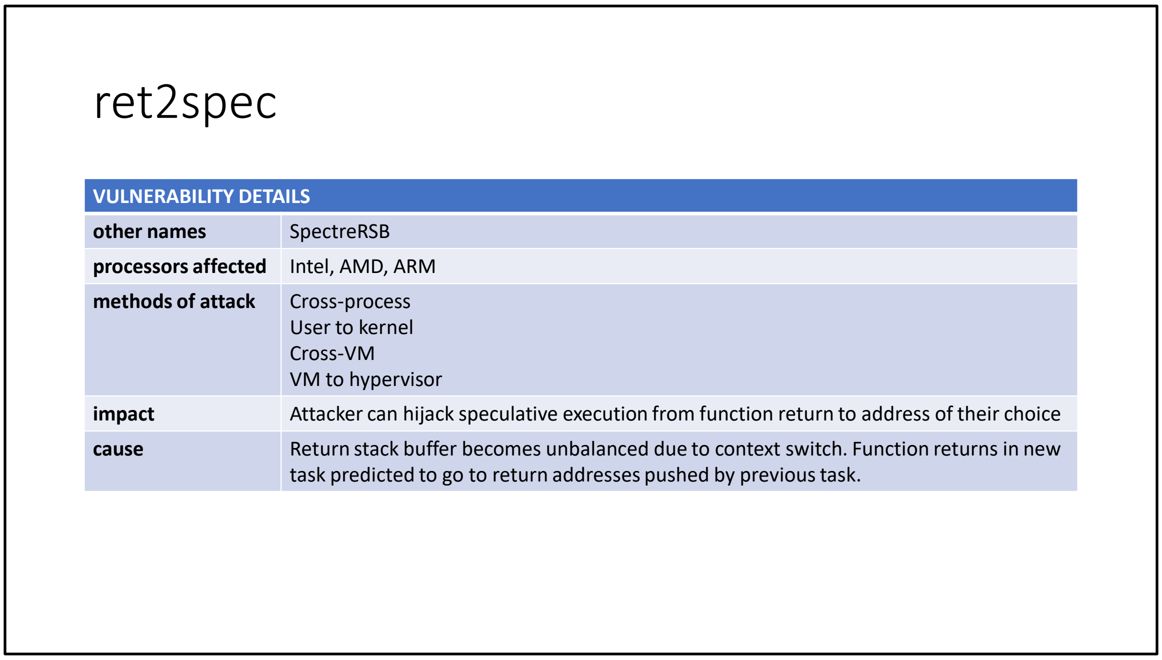
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5754
https://en.wikipedia.org/wiki/Meltdown_(security_vulnerability)
https://arxiv.org/pdf/1801.01207.pdf
https://software.intel.com/security-software-guidance/software-guidance/rogue-data-cache-load
https://gruss.cc/files/kaiser.pdf
https://msrc-blog.microsoft.com/2018/03/23/kva-shadow-mitigating-meltdown-on-windows/

meltdown – mitigation

KAISER, aka:
- kernel page table isolation (KPTI)
- kernel virtual address (KVA) shadow

Split process page tables into **user** and **kernel** views
- User doesn't see kernel (almost) at all
- Kernel sees user as NX

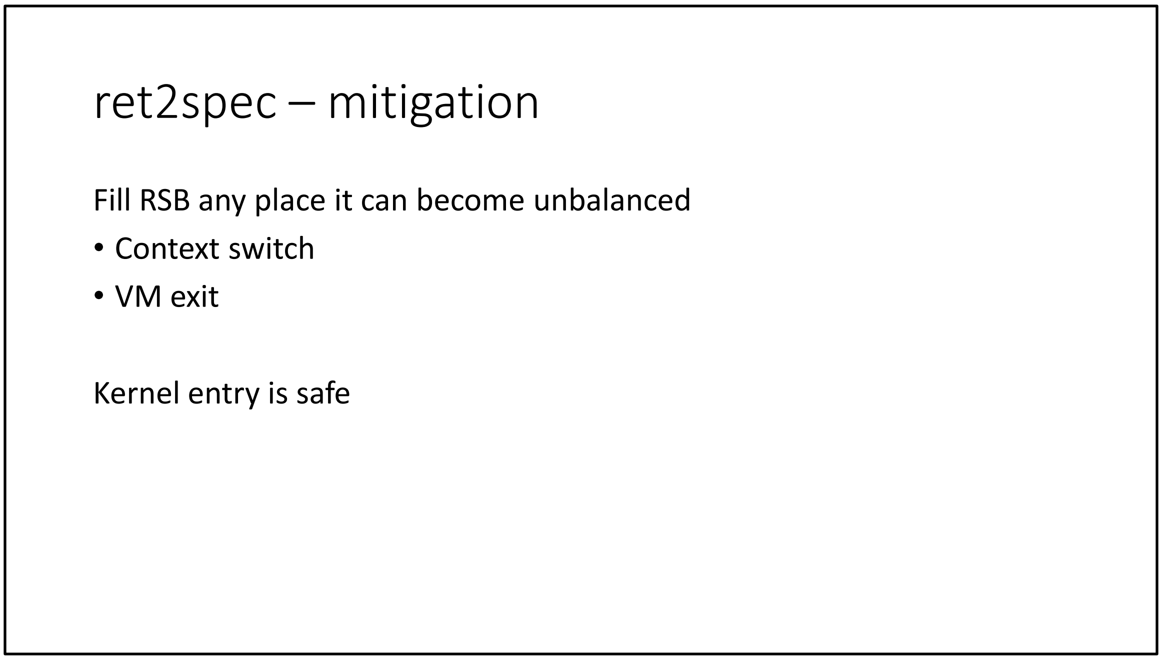
Requires PCID or ASID to avoid performance-killing TLB flushes



ret2spec

VULNERABILITY DETAILS	
other names	SpectreRSB
processors affected	Intel, AMD, ARM
methods of attack	Cross-process User to kernel Cross-VM VM to hypervisor
impact	Attacker can hijack speculative execution from function return to address of their choice
cause	Return stack buffer becomes unbalanced due to context switch. Function returns in new task predicted to go to return addresses pushed by previous task.

https://christian-rossow.de/publications/ret2spec-ccs2018.pdf
https://www.usenix.org/system/files/conference/woot18/woot18-paper-koruyeh.pdf



ret2spec – mitigation

Fill RSB any place it can become unbalanced

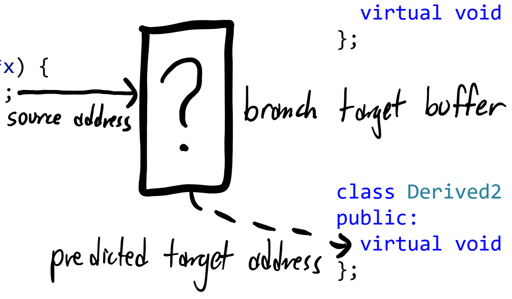
- Context switch
- VM exit

Kernel entry is safe

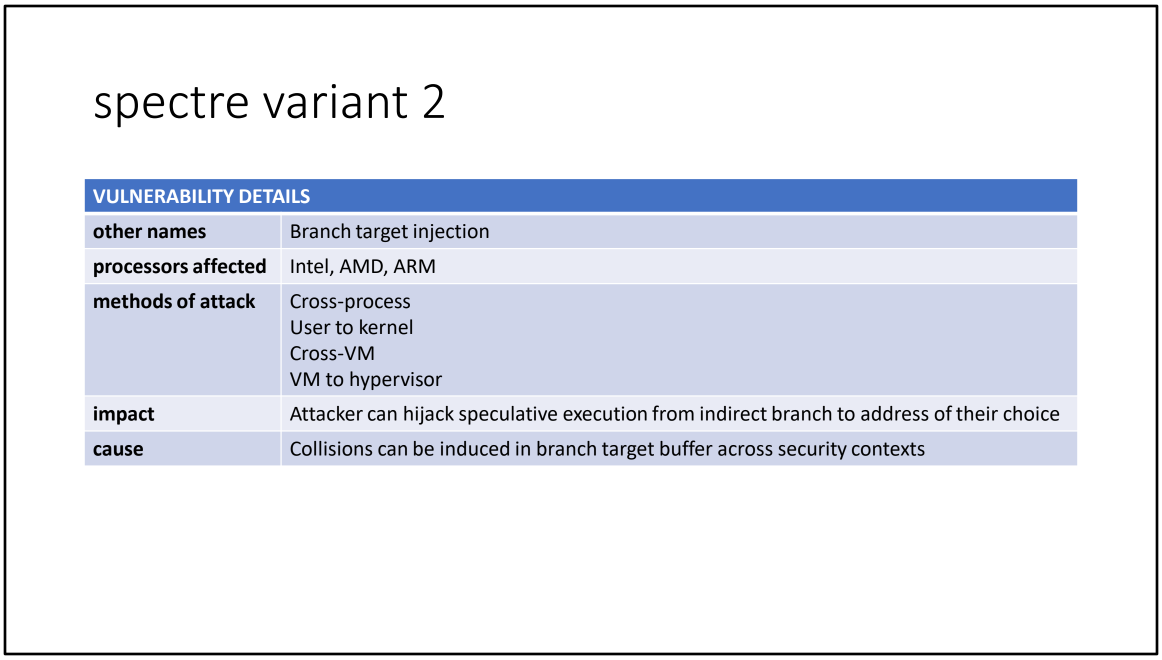
interlude: indirect branch prediction

```
class Derived1 : public Base {
public:
  virtual void Method() {}
};
```

```
void f(Base *x) {
  x->Method();
}
```



```
class Derived2 : public Base {
public:
  virtual void Method() {}
};
```



spectre variant 2

VULNERABILITY DETAILS	
other names	Branch target injection
processors affected	Intel, AMD, ARM
methods of attack	Cross-process User to kernel Cross-VM VM to hypervisor
impact	Attacker can hijack speculative execution from indirect branch to address of their choice
cause	Collisions can be induced in branch target buffer across security contexts

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5715
https://support.google.com/faqs/answer/7625886
https://software.intel.com/security-software-guidance/insights/deep-dive-retpoline-branch-target-injection-mitigation
https://software.intel.com/security-software-guidance/insights/deep-dive-indirect-branch-restricted-speculation
https://software.intel.com/security-software-guidance/insights/deep-dive-single-thread-indirect-branch-predictors
https://developer.amd.com/wp-content/resources/Architecture_Guidelines_Update_Indirect_Branch_Control.pdf
https://trustedfirmware-a.readthedocs.io/en/latest/security_advisories/security-advisory-tfv-6.htm

spectre variant 2 – mitigation (x86)

retpoline

```
# Implement indirect branch to r11
__llvm_retpoline_r11:
    call    .L2                    # push .L1 on RSB
.L1:
    pause                          # annotate spinlock
    lfence                         # try to end speculation
    jmp     .L1                    # loop forever
.L2:
    mov     qword ptr [rsp], r11   # overwrite return address
    ret                            # send speculation to .L1
                                   # (real execution goes to r11)
```

spectre variant 2 – mitigation (x86)

using retpoline

`clang kernel.cc` **`-mretpoline`**

transforms

```
call     qword ptr [rcx]
```

into

```
mov      r11, qword ptr [rcx]
call     __llvm_retpoline_r11
```

spectre variant 2 – mitigation (x86)

cross-process and cross-vm protection

Indirect Branch Predictor Barrier (IBPB)

Clear predictor state for logical processor; used on process/VM switch

Single Thread Indirect Branch Predictor (STIBP)

Prevent interference between threads on a core

spectre variant 2 – mitigation (x86)
doing better than retpoline

Indirect Branch Restricted Speculation (IBRS)

- Prevents code at lower privilege from interfering with indirect branch targets predicted by **higher** privilege that runs **after** the command
- Must be triggered on every switch to higher privilege
- **DOES NOT** protect across processes or VMs

spectre variant 2 – mitigation (x86)

doing better than “better than retpoline”

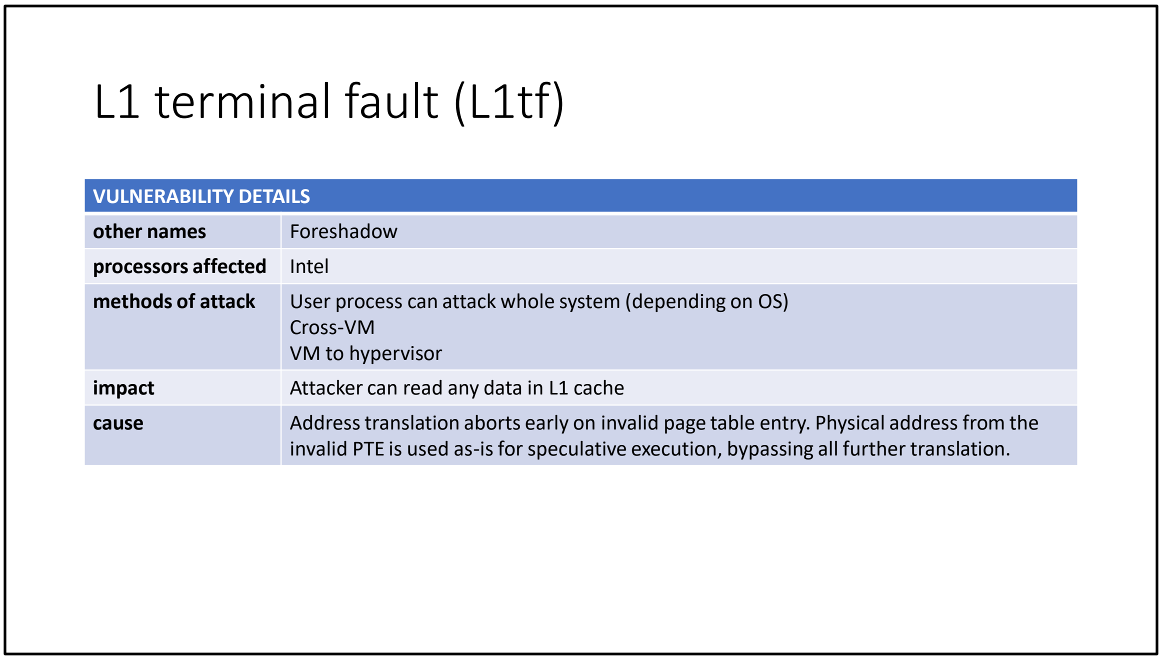
Enhanced IBRS

- “Sticky” IBRS – only needs to be enabled once
- Only safe if used with Supervisor-Mode Execution Prevention (SMEP)

spectre variant 2 – mitigation (arm)

- One big hammer: enter EL3, clear everything.
- Requires updated ARM Trusted Firmware
- Used on
  - Context switch
  - “exception entry from an exception level where it is judged that code might be used to attack a higher level” (?)

    Linux+Fuchsia: whenever a process faults trying to jump into kernel code
- Only safe if user code mapped “privileged execute never” (PXN)

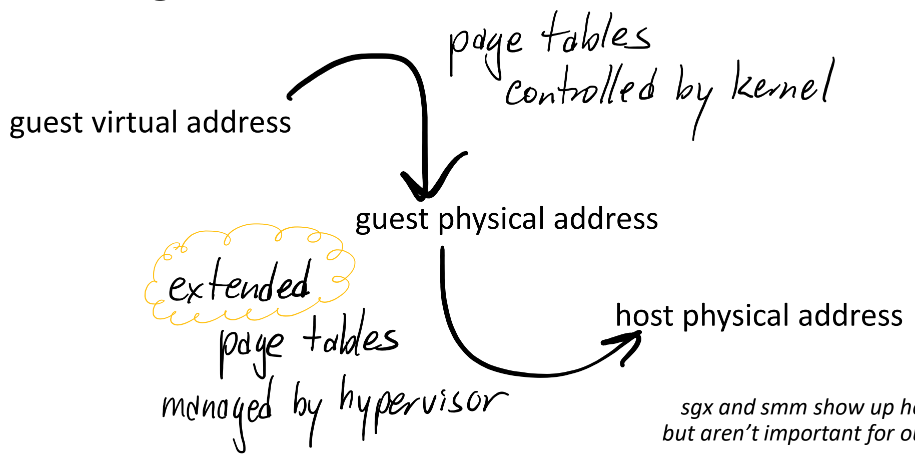


L1 terminal fault (L1tf)

VULNERABILITY DETAILS	
other names	Foreshadow
processors affected	Intel
methods of attack	User process can attack whole system (depending on OS) Cross-VM VM to hypervisor
impact	Attacker can read any data in L1 cache
cause	Address translation aborts early on invalid page table entry. Physical address from the invalid PTE is used as-is for speculative execution, bypassing all further translation.

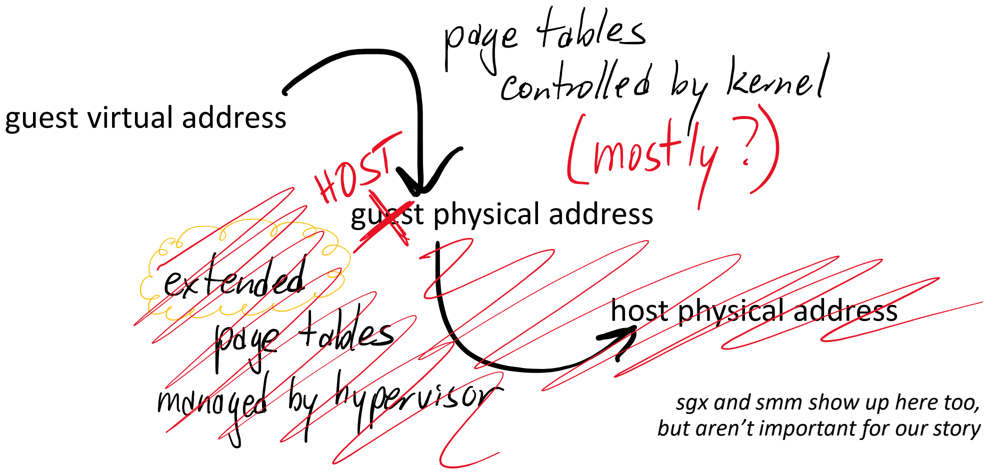
https://software.intel.com/security-software-guidance/insights/deep-dive-intel-analysis-l1-terminal-fault
https://docs.microsoft.com/en-us/virtualization/community/team-blog/2018/20180814-hyper-v-hyperclear-mitigation-for-l1-terminal-fault

nothing that more indirection can't solve


guest virtual address
page tables controlled by kernel
guest physical address
extended page tables managed by hypervisor
host physical address

sgx and smm show up here too,
but aren't important for our story

indirection indiscretion!


guest virtual address
page tables controlled by kernel
(mostly?)
HOST
~~guest~~ physical address
extended page tables managed by hypervisor
host physical address

sgx and smm show up here too,
but aren't important for our story

L1 terminal fault (L1tf) – mitigation

Against hostile processes

Don’t leave a valid physical address in invalid PTEs

Against hostile VMs

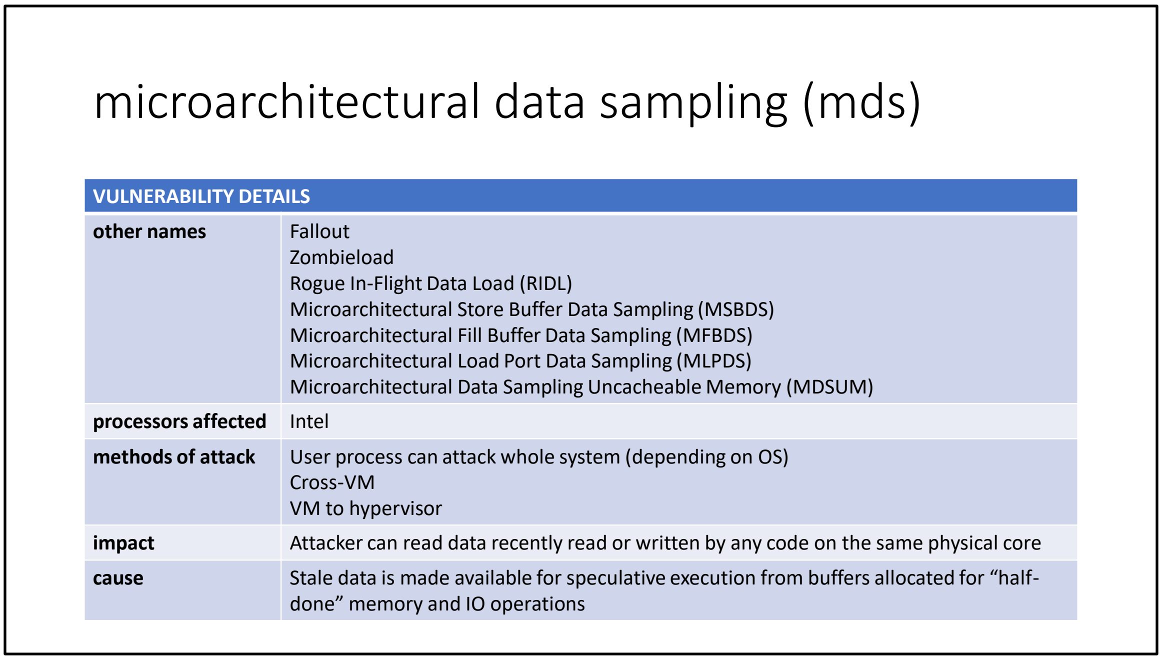
Don’t leave anything in L1D to leak

L1 terminal fault (L1tf) – mitigation

Against hostile VM coscheduled on the same physical core

Nope. Have to keep that from happening.

- Disable SMT
- Core scheduling



microarchitectural data sampling (mds)

VULNERABILITY DETAILS	
other names	Fallout Zombieload Rogue In-Flight Data Load (RIDL) Microarchitectural Store Buffer Data Sampling (MSBDS) Microarchitectural Fill Buffer Data Sampling (MFBDS) Microarchitectural Load Port Data Sampling (MLPDS) Microarchitectural Data Sampling Uncacheable Memory (MDSUM)
processors affected	Intel
methods of attack	User process can attack whole system (depending on OS) Cross-VM VM to hypervisor
impact	Attacker can read data recently read or written by any code on the same physical core
cause	Stale data is made available for speculative execution from buffers allocated for “half-done” memory and IO operations

https://software.intel.com/security-software-guidance/insights/deep-dive-intel-analysis-microarchitectural-data-sampling
https://googleprojectzero.blogspot.com/2020/02/escaping-chrome-sandbox-with-ridl.html
https://twitter.com/cpuGoogle/status/1254178356322398208?s=20

mds – mitigation

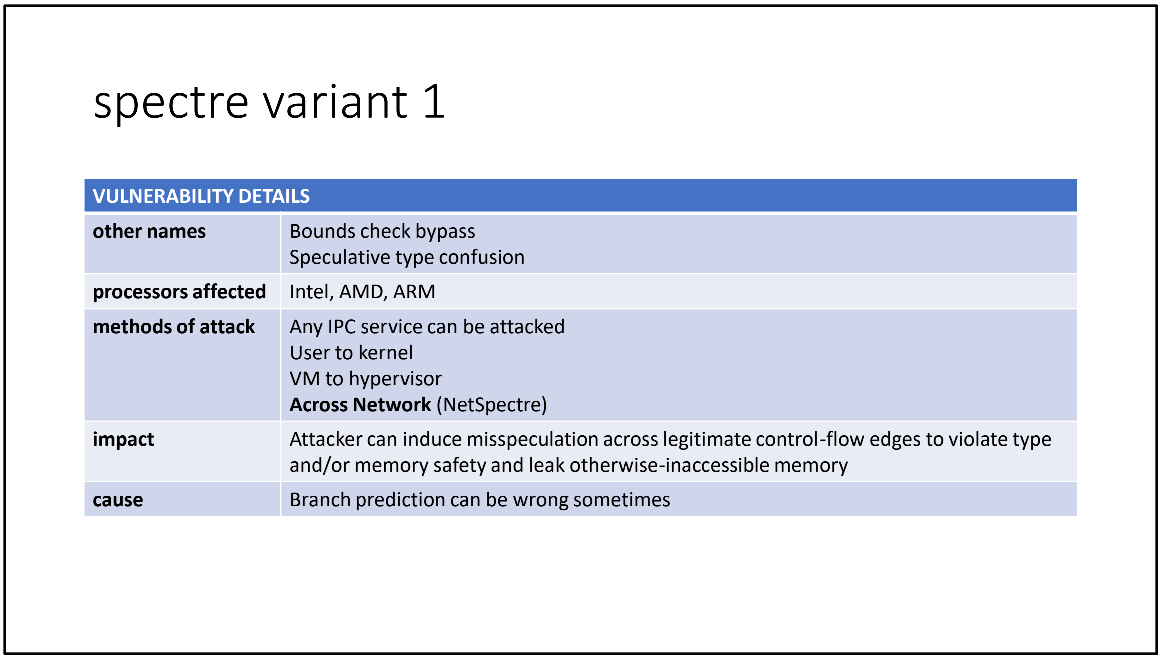
Against hostile processes or VMs

Newly magic `VERW` instruction clears buffers before returning control

Against hostile processes or VMs on adjacent hyperthread

Nope. Disable SMT or use core scheduling[1].

[1]nobody does this for processes. yet.



spectre variant 1

VULNERABILITY DETAILS	
other names	Bounds check bypass Speculative type confusion
processors affected	Intel, AMD, ARM
methods of attack	Any IPC service can be attacked User to kernel VM to hypervisor **Across Network** (NetSpectre)
impact	Attacker can induce misspeculation across legitimate control-flow edges to violate type and/or memory safety and leak otherwise-inaccessible memory
cause	Branch prediction can be wrong sometimes

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5753
https://llvm.org/docs/SpeculativeLoadHardening.html
https://arxiv.org/abs/1905.10311

spectre variant 1 – example

```c
int GetBufferHeader(int user_provided_buffer_index) {
  if (user_provided_buffer_index >= num_buffers) {
    return -ENOWAY;
  }

  // Buffers are allocated as pages
  int page_index =
      page_index_from_buffer[user_provided_buffer_index];
  return *(page_base + page_index * PAGE_SIZE);
}
```

spectre variant 1 – mitigation

```
// (x < y) ? a : b
static inline size_t conditional_select_nospec_lt(
    size_t x, size_t y, size_t a, size_t b) {
  size_t select = a;     // Choose `a` to start

  __asm__(
      "cmp %2, %1\n"     // Compare `x` and `y`
      "cmovae %3, %0\n"  // if `x` >= `y`, choose `b` instead
      : "+r"(select)
      : "r"(x), "r"(y), "r"(b)
      : "cc");

  return select;
}
```

spectre variant 1 – mitigation example

```
int GetBufferHeaderSafe(int user_provided_buffer_index) {
  if (user_provided_buffer_index >= num_buffers) {
    return -ENOWAY;
  }

  // safe_index = (x < y) ? a : b
  int safe_index = conditional_select_nospec_lt(user_provided_buffer_index,   // x
                                               num_buffers,                  // y
                                               user_provided_buffer_index,   // a
                                               0);                           // b

  // Buffers are allocated as pages
  int page_index = page_index_from_buffer[safe_index];
  return *(page_base + page_index * PAGE_SIZE);
}
```

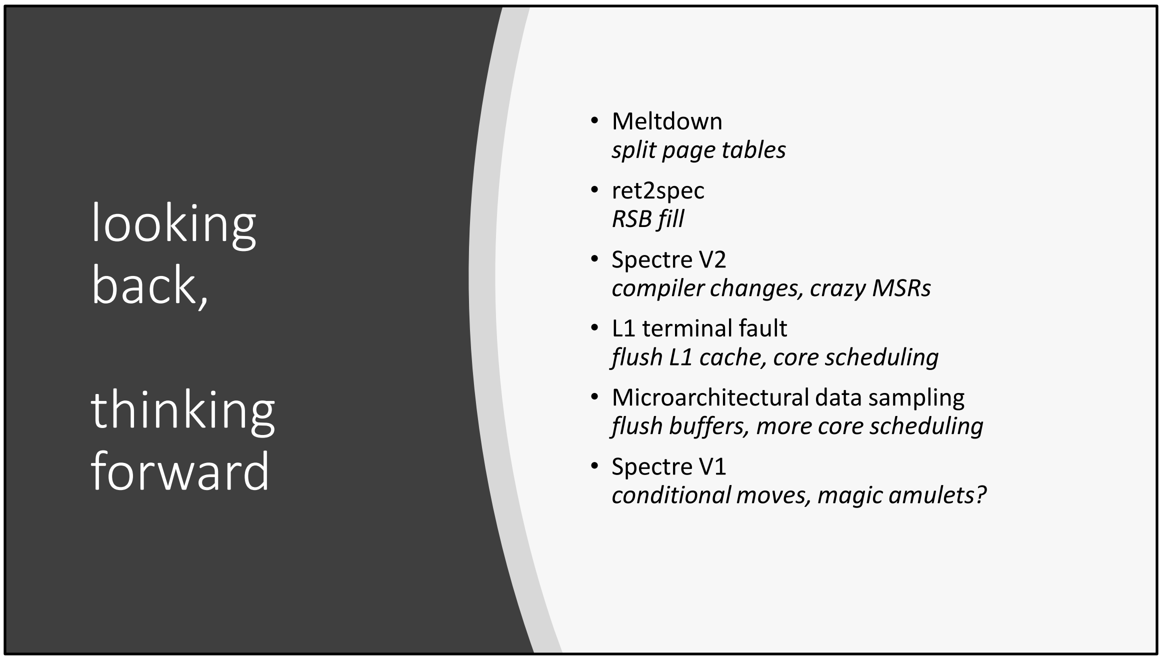
good LFENCEs make good neighbors

Speculation barrier

Wait for all prior instructions to complete before any subsequent instruction can begin executing.

x86	`LFENCE` (requires an MSR tweak on AMD)
ARM	`DSB; ISB` (anywhere) `SB` (ARMv8.5+)

https://github.com/google/safeside/blob/main/docs/fencing.md

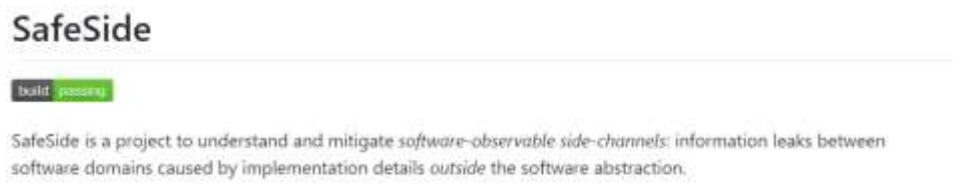


looking back,

thinking forward

- Meltdown
  split page tables
- ret2spec
  RSB fill
- Spectre V2
  compiler changes, crazy MSRs
- L1 terminal fault
  flush L1 cache, core scheduling
- Microarchitectural data sampling
  flush buffers, more core scheduling
- Spectre V1
  conditional moves, magic amulets?

putting our mitigations to the test(s)



SafeSide

build passing

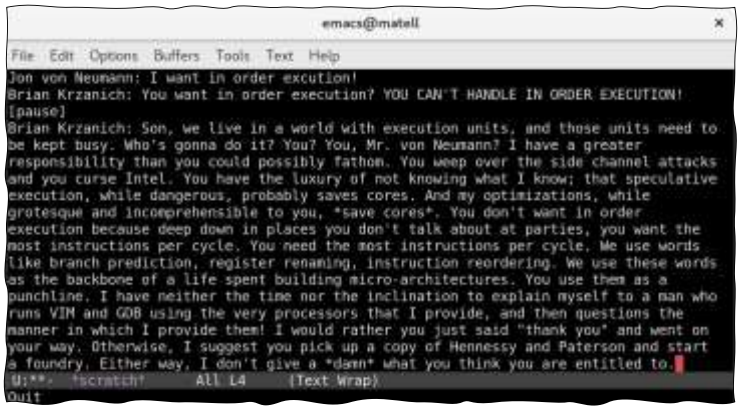
SafeSide is a project to understand and mitigate *software-observable side-channels*: information leaks between software domains caused by implementation details *outside* the software abstraction.

`github.com/google/`**`safeside`**

wrapping up

keeping up

- Read everything, from everyone
- Look at Linux
- Test!



```
Jon von Neumann: I want in order excution!
Brian Krzanich: You want in order execution? YOU CAN'T HANDLE IN ORDER EXECUTION!
[pause]
Brian Krzanich: Son, we live in a world with execution units, and those units need to
be kept busy. Who's gonna do it? You? You, Mr. von Neumann? I have a greater
responsibility than you could possibly fathom. You weep over the side channel attacks
and you curse Intel. You have the luxury of not knowing what I know; that speculative
execution, while dangerous, probably saves cores. And my optimizations, while
grotesque and incomprehensible to you, *save cores*. You don't want in order
execution because deep down in places you don't talk about at parties, you want the
most instructions per cycle. You need the most instructions per cycle. We use words
like branch prediction, register renaming, instruction reordering. We use these words
as the backbone of a life spent building micro-architectures. You use them as a
punchline. I have neither the time nor the inclination to explain myself to a man who
runs VIM and GDB using the very processors that I provide, and then questions the
manner in which I provide them! I would rather you just said "thank you" and went on
your way. Otherwise, I suggest you pick up a copy of Hennessy and Paterson and start
a foundry. Either way, I don't give a *damn* what you think you are entitled to.
U:**-  *scratch*      All L4     (Text Wrap)
Quit
```

twitter.com/ellism

Matthew Riley
mattdr@google.com
@mdriley25519
github.com/google/safeside

Thank you!

Resources that didn't fit on just one slide:

https://developer.arm.com/support/arm-security-updates/speculative-processor-vulnerability/download-the-whitepaper
https://developer.amd.com/wp-content/resources/90343-B_SoftwareTechniquesforManagingSpeculation_WP_7-18Update_FNL.pdf
https://www.kernel.org/doc/html/latest/admin-guide/hw-vuln/index.html