

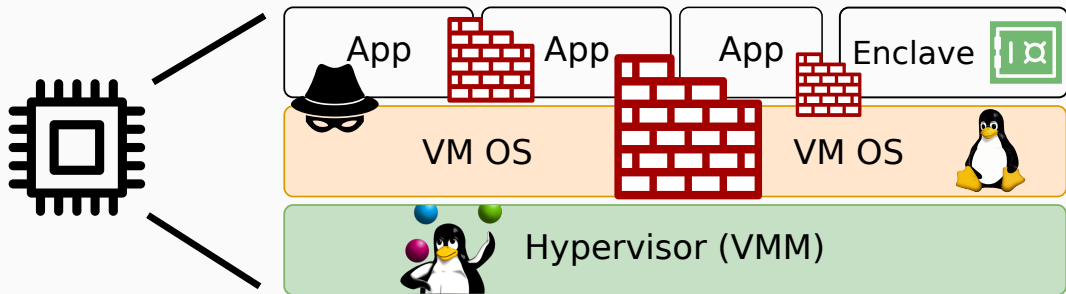
LVI

Hijacking Transient Execution with Load Value Injection

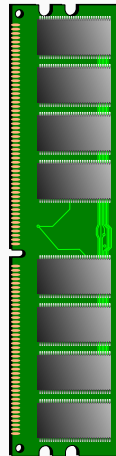
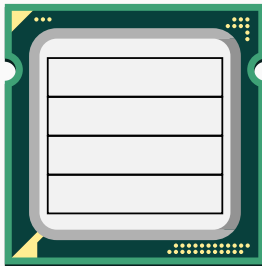
Daniel Gruss, Daniel Moghimi, Jo Van Bulck

Hardwear.io Virtual Con, April 30, 2020



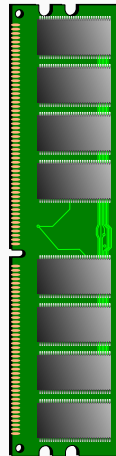
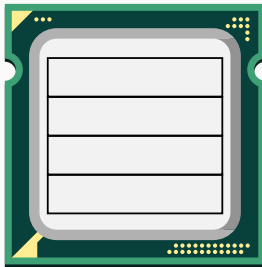


```
printf("%d", i);  
printf("%d", i);
```



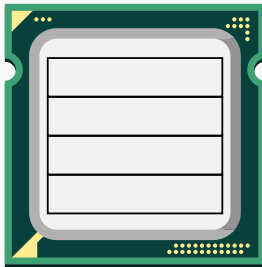
```
printf("%d", i);  
printf("%d", i);
```

Cache miss

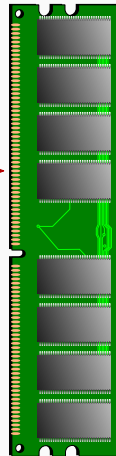


```
printf("%d", i);  
printf("%d", i);
```

Cache miss

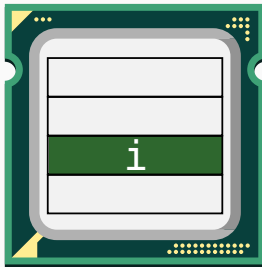


Request



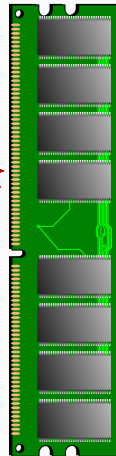
```
printf("%d", i);  
printf("%d", i);
```

Cache miss



Request

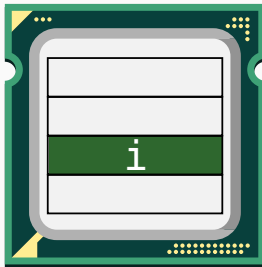
Response



```
printf("%d", i);  
printf("%d", i);
```

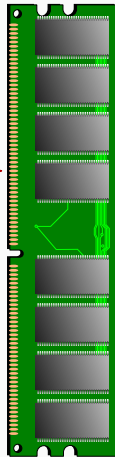
Cache miss

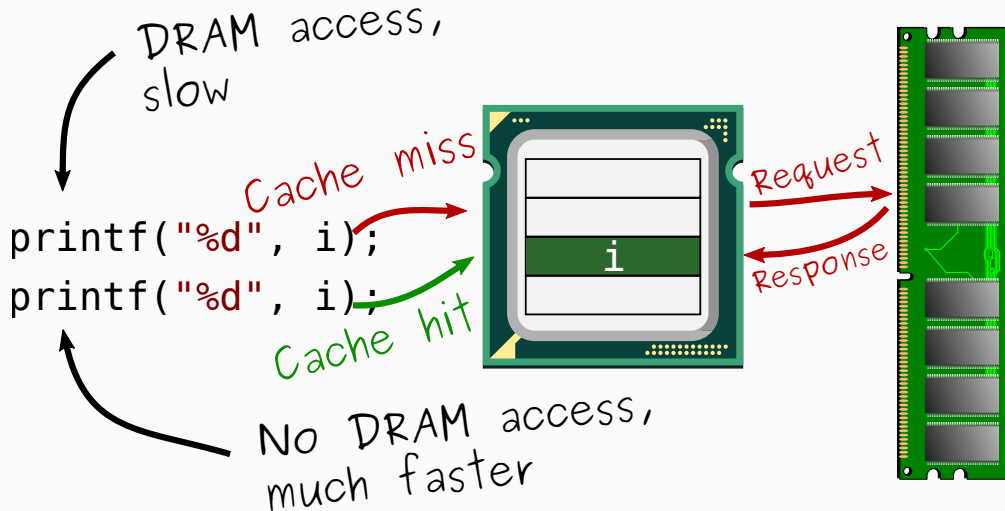
Cache hit



Request

Response

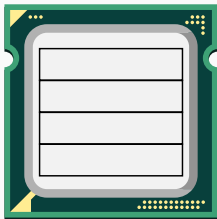




Shared Memory

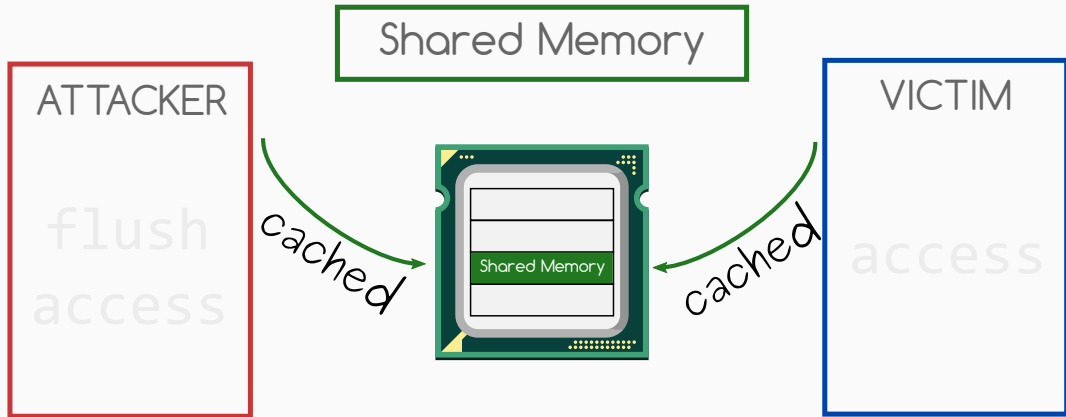
ATTACKER

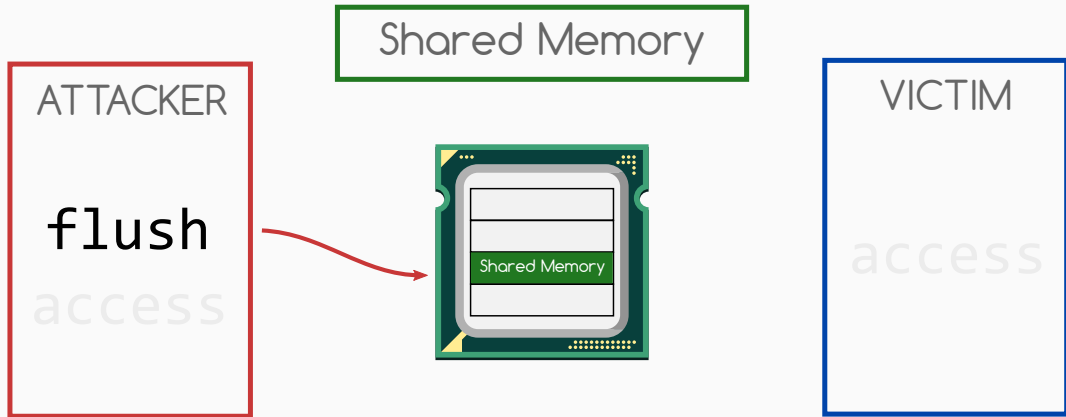
flush
access

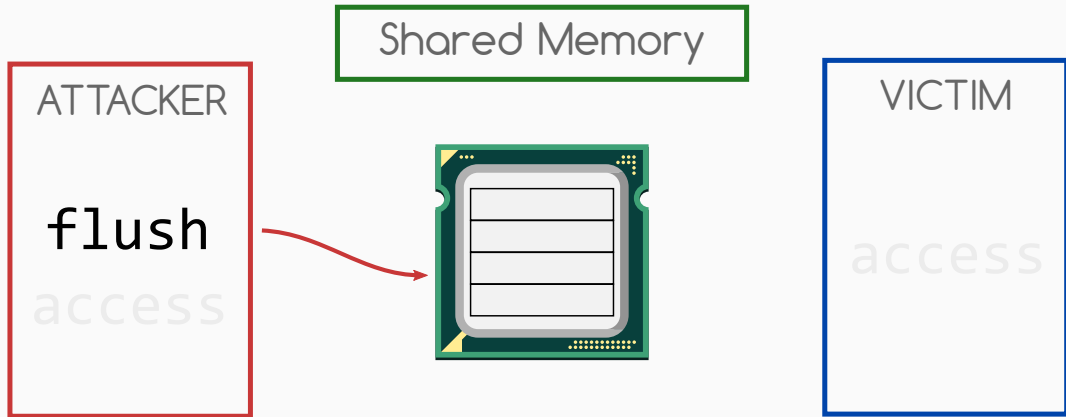


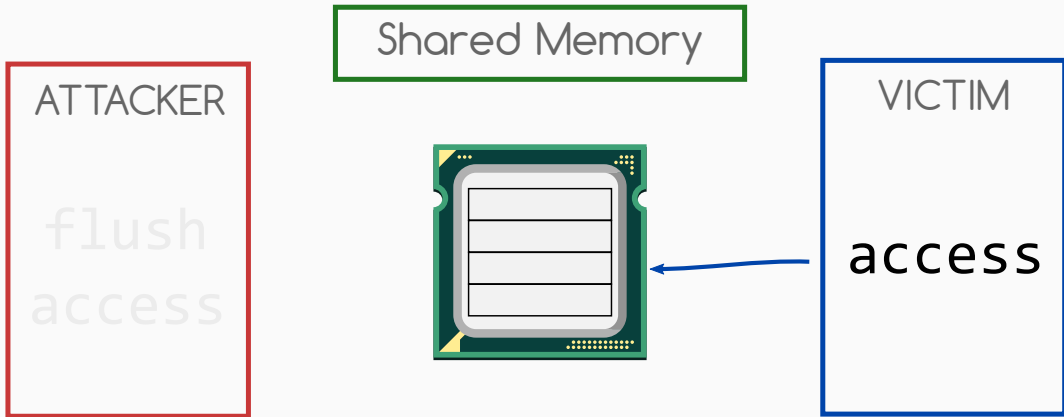
VICTIM

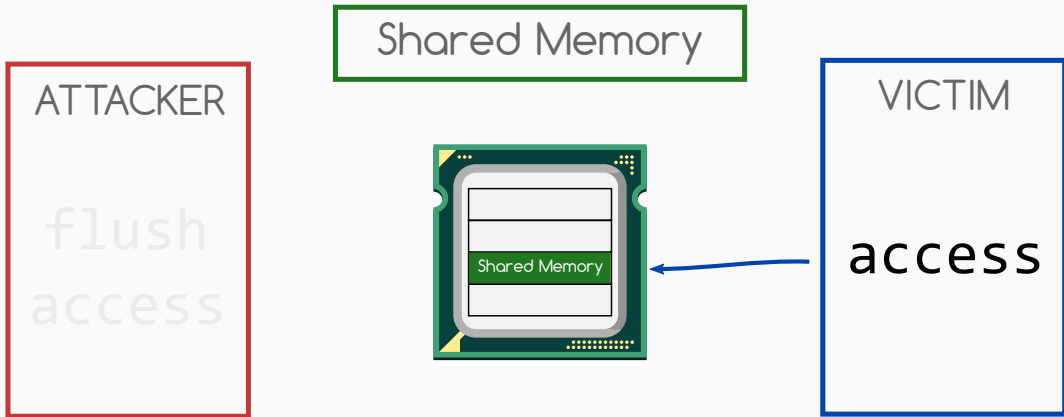
access

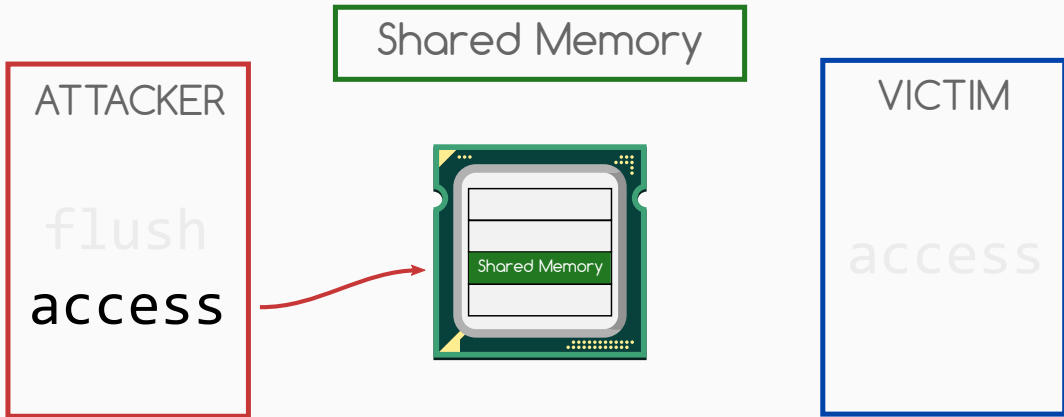


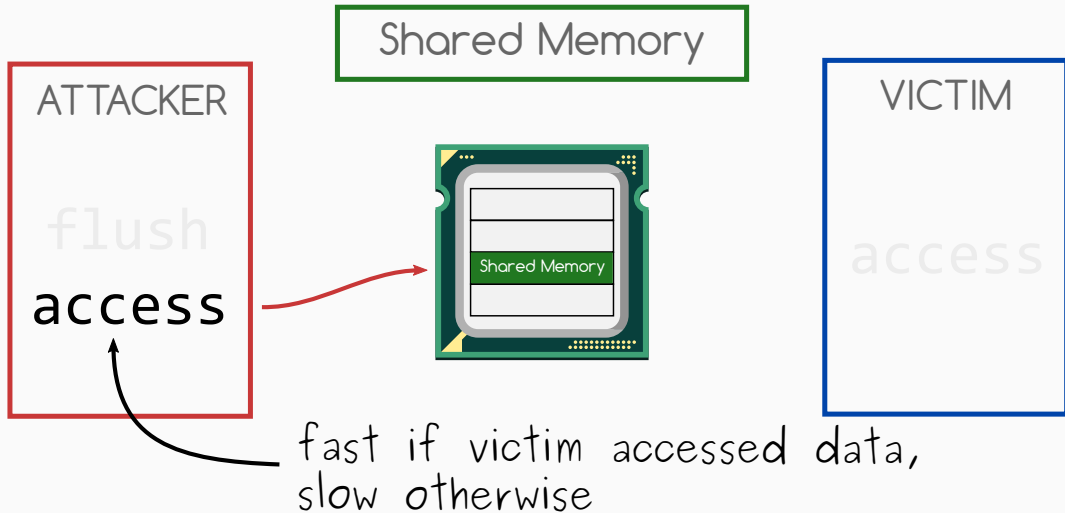


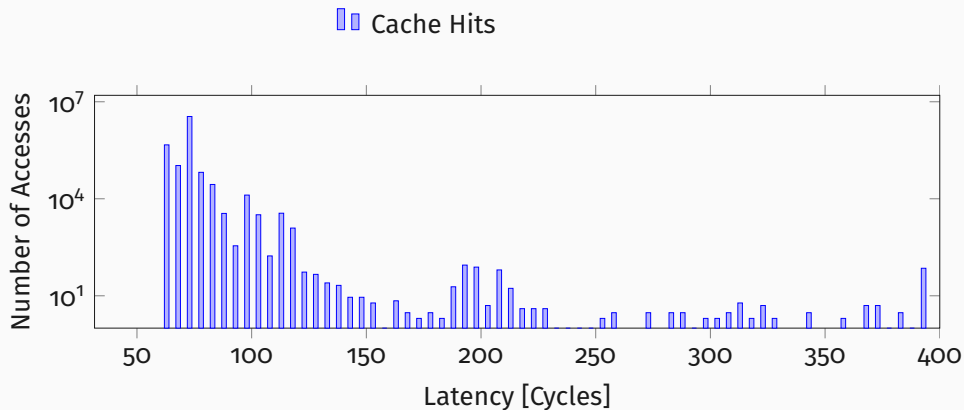


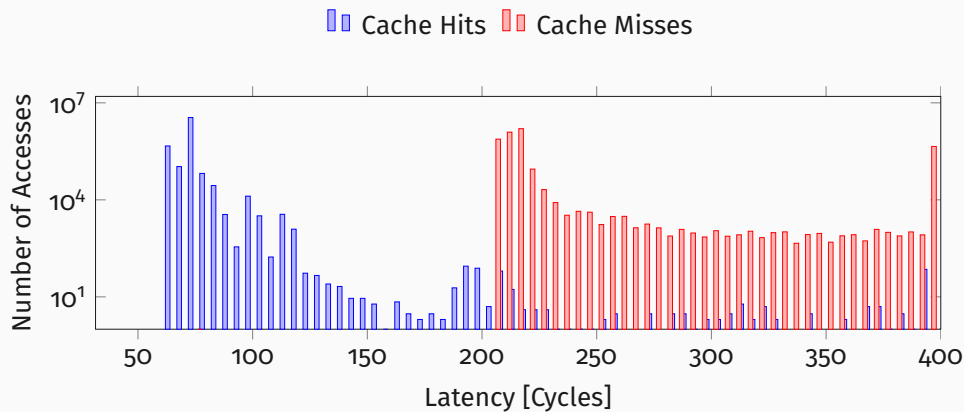


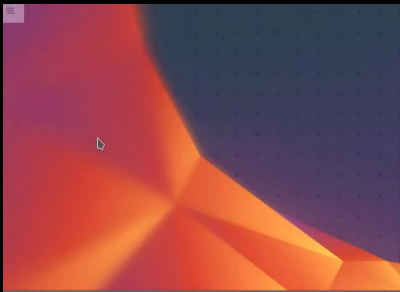








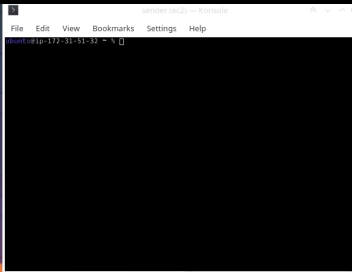




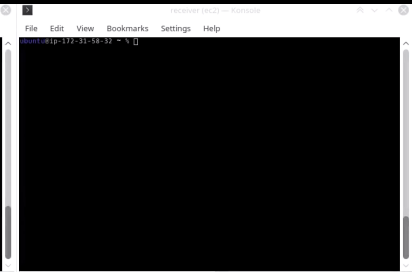
local — Konsole

File Edit View Bookmarks Settings Help

michael@michael-hp ~ %



sender (ec2)



receiver (ec2)

sender (ec2) — Konsole <2>

File Edit View Bookmarks Settings Help

-- slurm 0.4.3 on ip-172-31-51-32 --

```

x
x
xx
xxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx
xx
x
x

Active Interface: eth0          Interface Speed: unknown
Current RX Speed: 0.05 KB/s      Current TX Speed: 0.33 KB/s
Graph Top RX Speed: 0.98 KB/s    Graph Top TX Speed: 2.64 KB/s
Overall Top RX Speed: 0.98 KB/s  Overall Top TX Speed: 2.64 KB/s
Received Packets: 84             Transmitted Packets: 82
MBytes Received: 0.005 MB        MBytes Transmitted: 0.019 MB
Errors on Receiving: 0           Errors on Transmission: 0
```

sender (ec2)

receiver (ec2) — Konsole <2>

File Edit View Bookmarks Settings Help

-- slurm 0.4.3 on ip-172-31-58-32 --

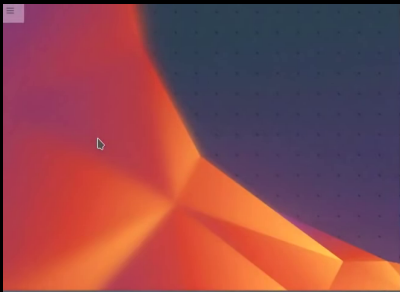
```

x
x
xx
xxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx
xx
x
x

Active Interface: eth0          Interface Speed: unknown
Current RX Speed: 0.05 KB/s      Current TX Speed: 0.29 KB/s
Graph Top RX Speed: 0.36 KB/s    Graph Top TX Speed: 0.84 KB/s
Overall Top RX Speed: 0.36 KB/s  Overall Top TX Speed: 0.84 KB/s
Received Packets: 38             Transmitted Packets: 36
MBytes Received: 0.002 MB        MBytes Transmitted: 0.010 MB
Errors on Receiving: 0           Errors on Transmission: 0
```

receiver (ec2)

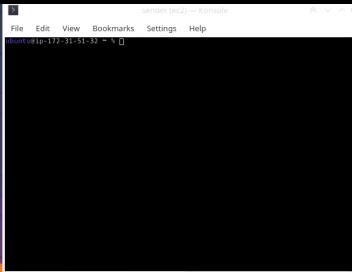
HELLO FROM THE OTHER SIDE (DEMO): VIDEO STREAMING OVER CACHE COVERT CHANNEL



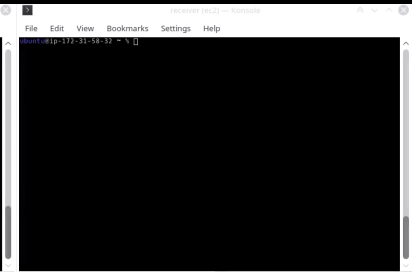
local — Konsole

File Edit View Bookmarks Settings Help

michael@michael-hp ~ %



sender (ec2)



receiver (ec2)

sender (ec2) — Konsole <2>

File Edit View Bookmarks Settings Help

-- slurm 0.4.3 on ip-172-31-51-32 --

```

x
x
xx
xxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx
xx
x
x

Active Interface: eth0          Interface Speed: unknown
Current RX Speed: 0.05 KB/s      Current TX Speed: 0.33 KB/s
Graph Top RX Speed: 0.98 KB/s    Graph Top TX Speed: 2.64 KB/s
Overall Top RX Speed: 0.98 KB/s  Overall Top TX Speed: 2.64 KB/s
Received Packets: 84             Transmitted Packets: 82
MBytes Received: 0.005 MB        MBytes Transmitted: 0.019 MB
Errors on Receiving: 0           Errors on Transmission: 0
```

sender (ec2)

receiver (ec2) — Konsole <2>

File Edit View Bookmarks Settings Help

-- slurm 0.4.3 on ip-172-31-58-32 --

```

x
x
xx
xxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
xx
xx
x
x

Active Interface: eth0          Interface Speed: unknown
Current RX Speed: 0.05 KB/s      Current TX Speed: 0.29 KB/s
Graph Top RX Speed: 0.36 KB/s    Graph Top TX Speed: 0.84 KB/s
Overall Top RX Speed: 0.36 KB/s  Overall Top TX Speed: 0.84 KB/s
Received Packets: 38             Transmitted Packets: 36
MBytes Received: 0.002 MB        MBytes Transmitted: 0.010 MB
Errors on Receiving: 0           Errors on Transmission: 0
```

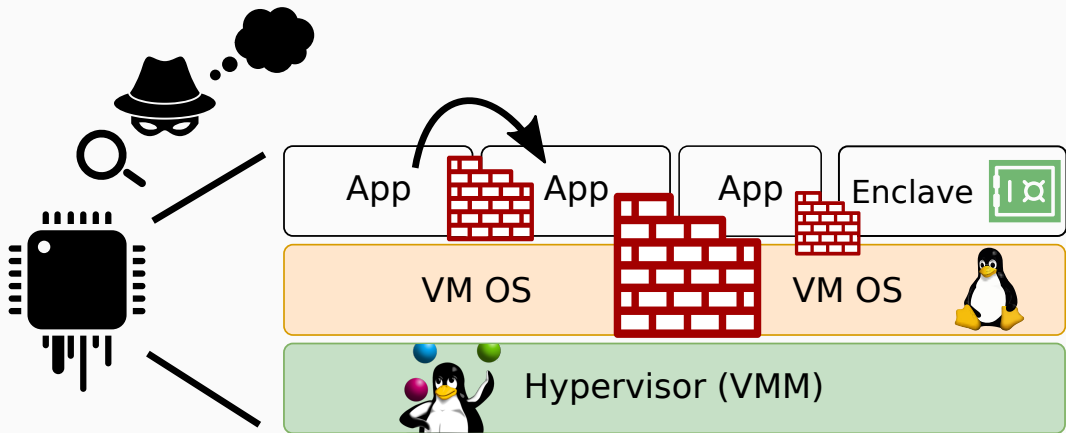
receiver (ec2)

HELLO FROM THE OTHER SIDE (DEMO): VIDEO STREAMING OVER CACHE COVERT CHANNEL



**We can communicate across protection walls
using microarchitectural side-channels!**

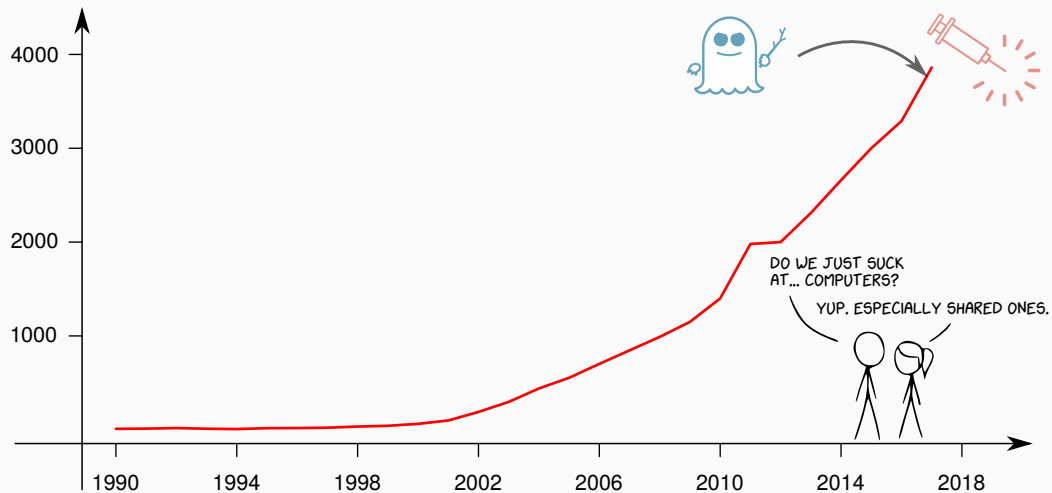
Leaky processors: Jumping over protection walls with side-channels



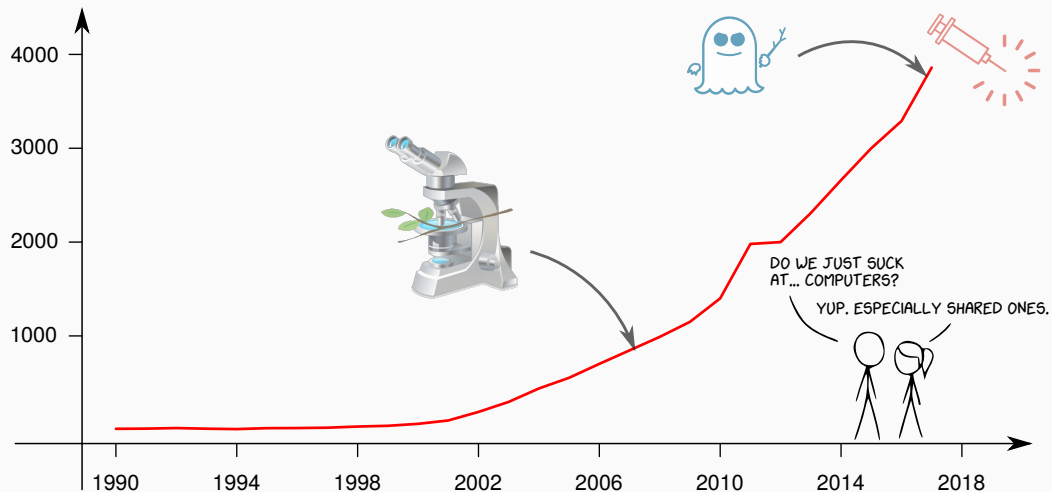
SHARING IS NOT CARING

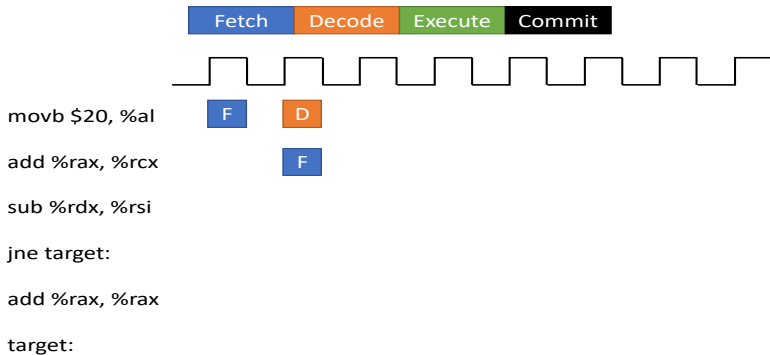
SHARING IS LOSING YOUR STUFF TO OTHERS

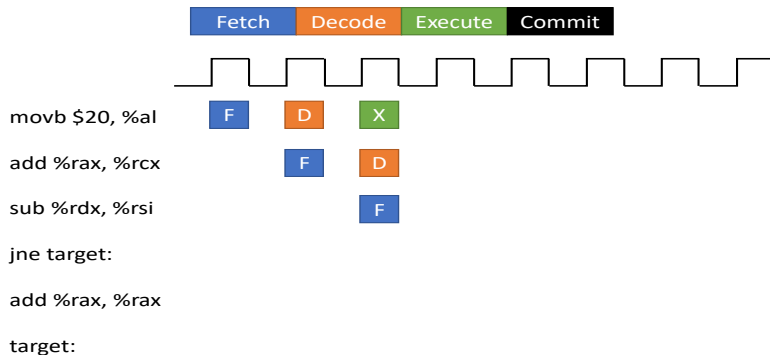
Side-channel attacks are known for decades already – what's new?



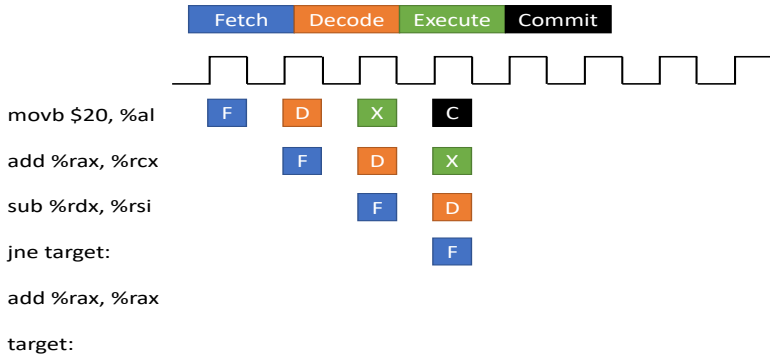
Side-channel attacks are known for decades already – what's new?



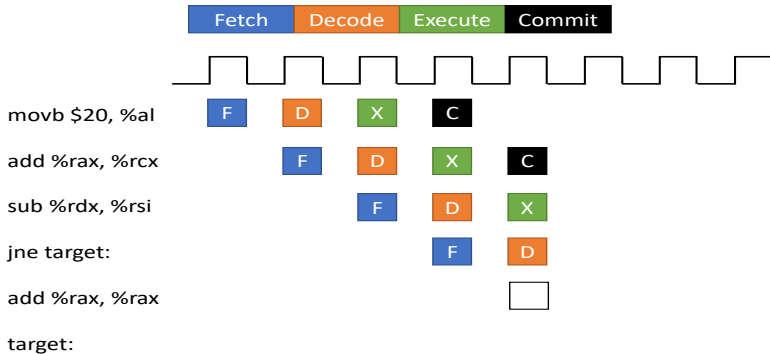




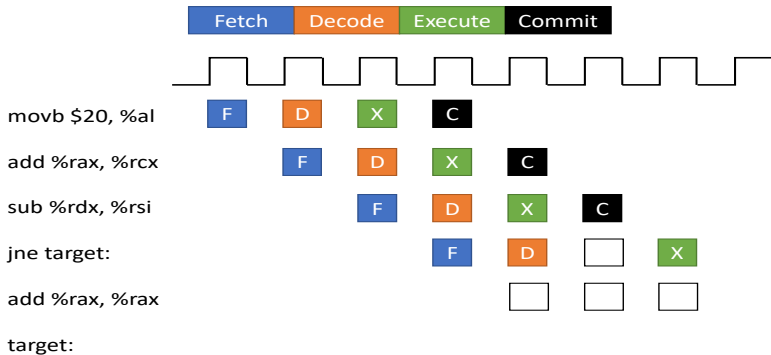
Pipeline Bubble



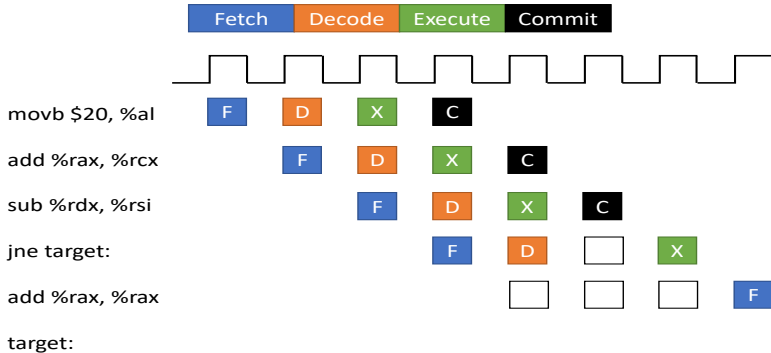
Pipeline Bubble



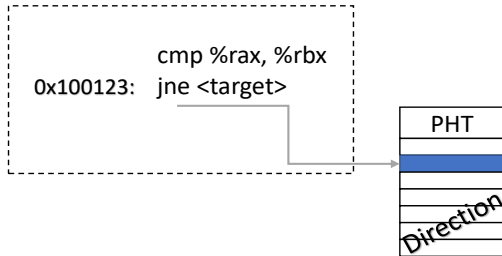
Pipeline Bubble

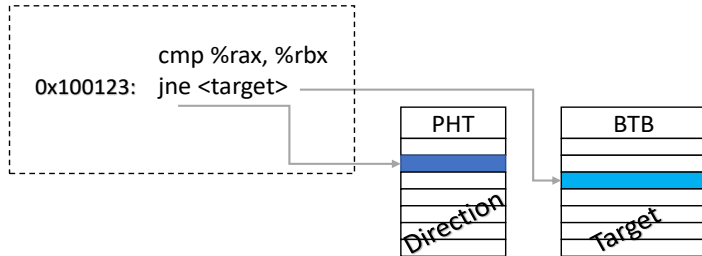


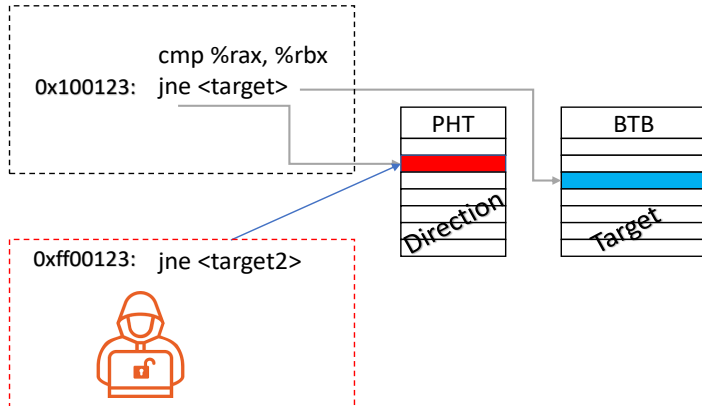
Pipeline Bubble

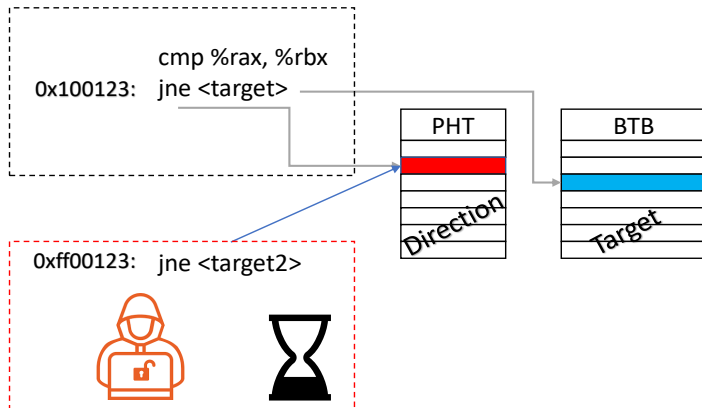


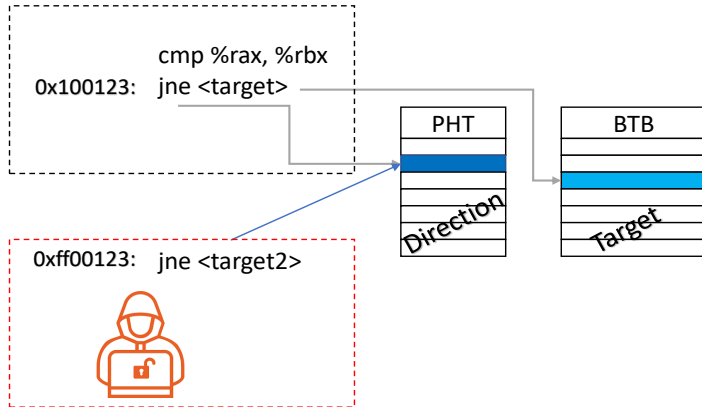

```
                cmp %rax, %rbx  
0x100123:  jne <target>
```

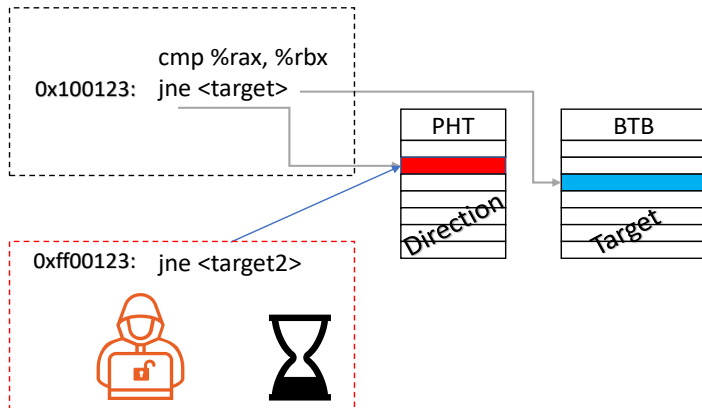




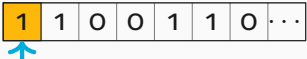




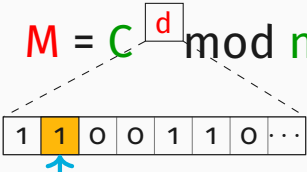




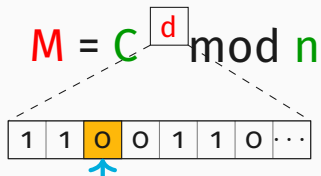
$$M = C^d \bmod n$$

$$M = C^d \bmod n$$


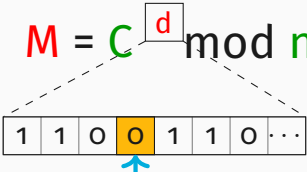
Result = C

$$M = C^d \bmod n$$


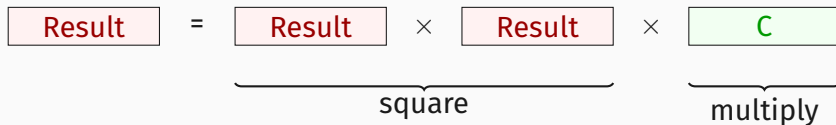
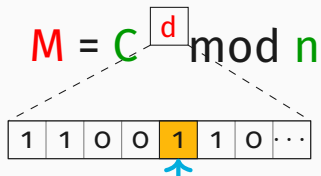
$$\text{Result} = \underbrace{\text{Result} \times \text{Result}}_{\text{square}} \times \underbrace{C}_{\text{multiply}}$$

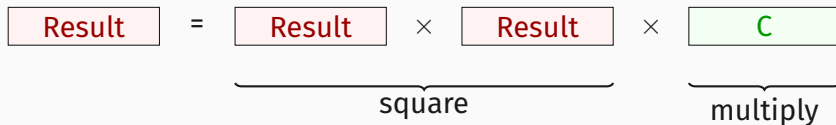
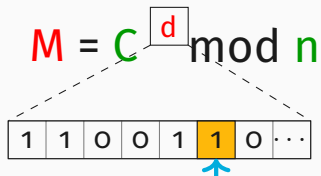


$$\text{Result} = \underbrace{\text{Result} \times \text{Result}}_{\text{square}}$$

$$M = C^d \bmod n$$


$$\text{Result} = \underbrace{\text{Result} \times \text{Result}}_{\text{square}}$$

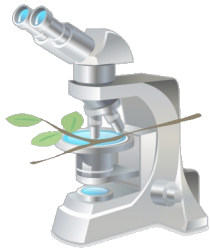




$$M = C^d \bmod n$$

1 1 0 0 1 1 0 ...

$$\text{Result} = \underbrace{\text{Result} \times \text{Result}}_{\text{square}}$$



?

?

?

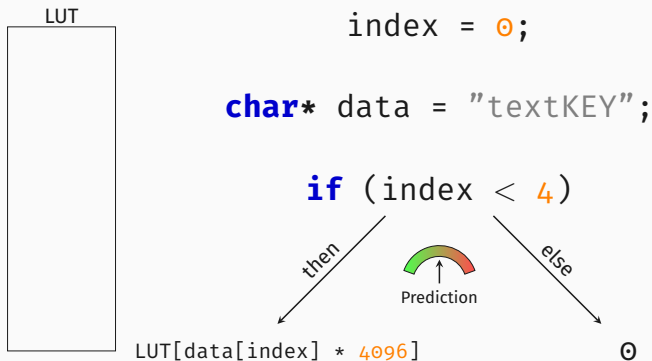
THE WHITE HOUSE
6:14 PM

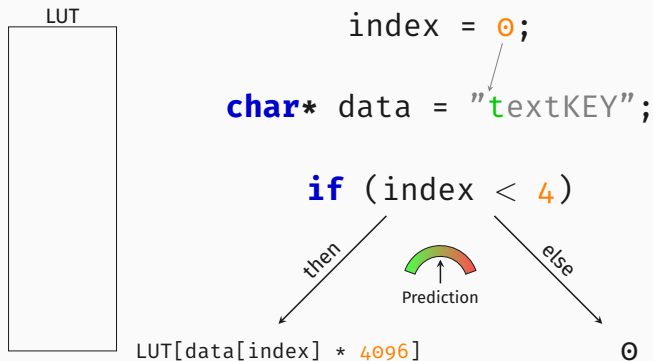


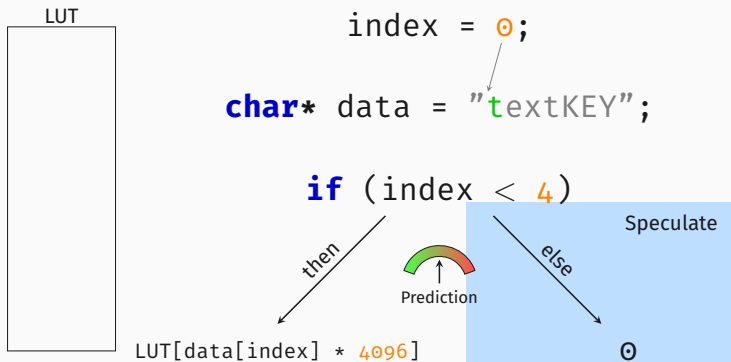
BREAKING NEWS

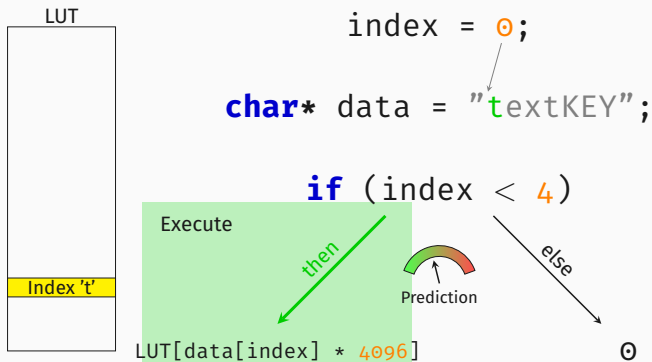
PRES. TRUMP UPDATES PUBLIC ON FEDERAL RESPONSE TO VIRUS

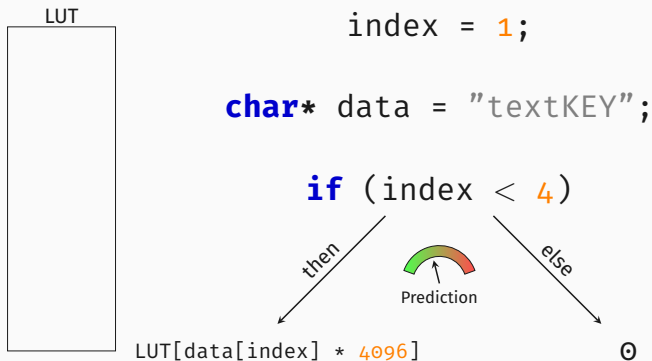
 **MSNBC**

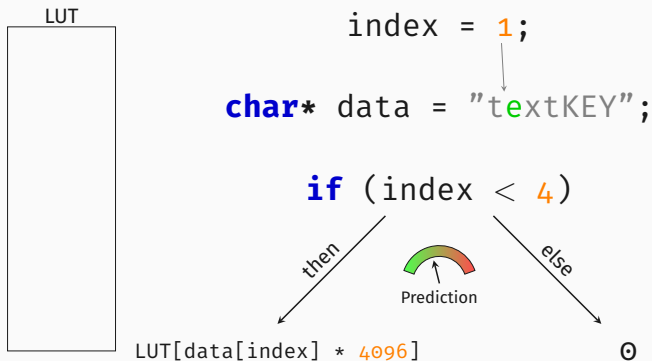


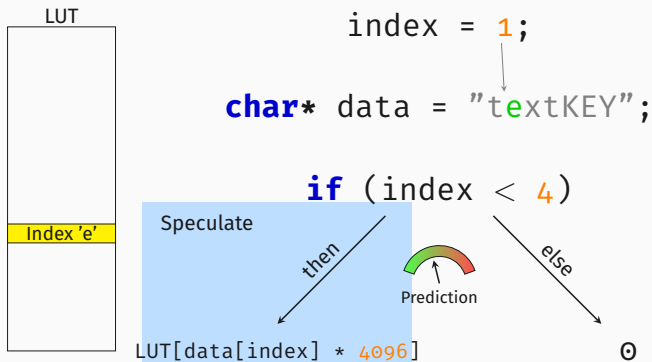


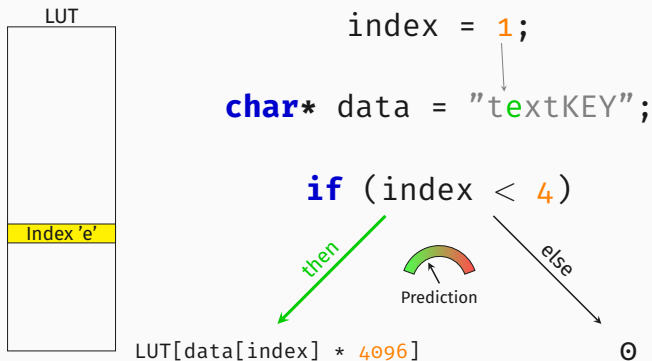


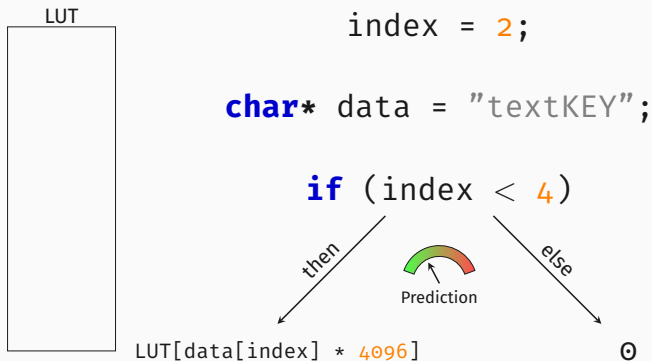


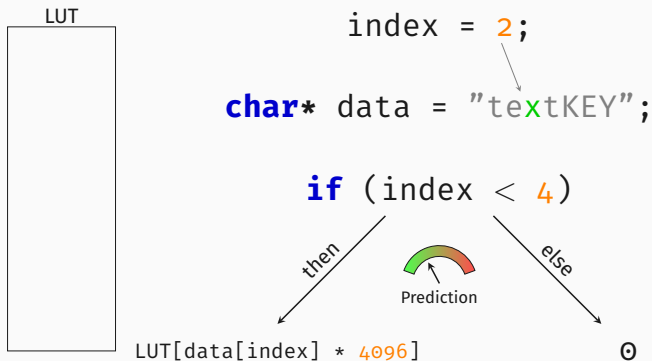


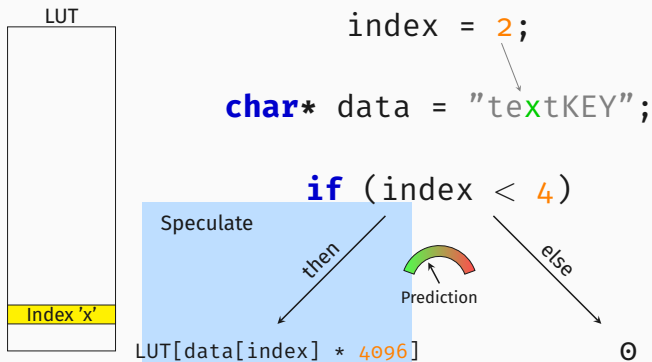


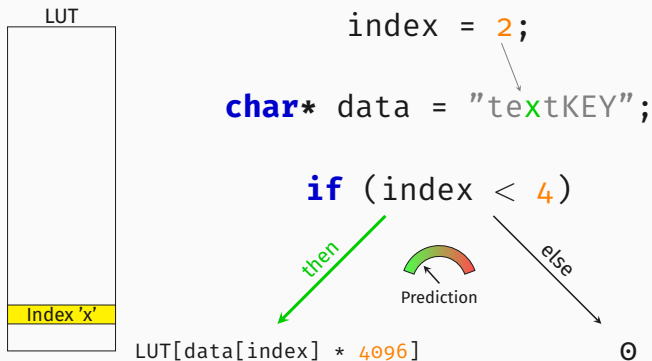


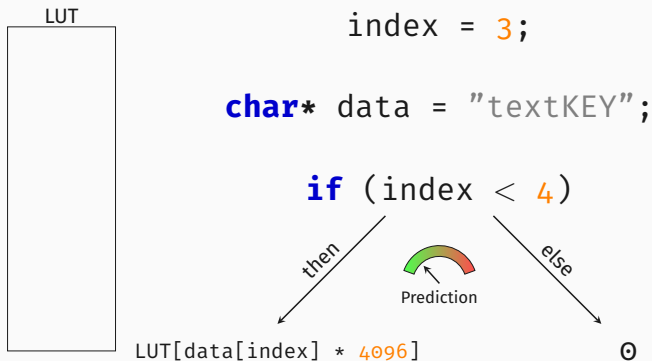


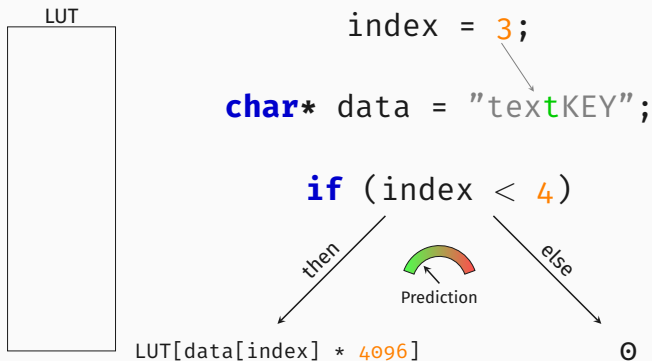


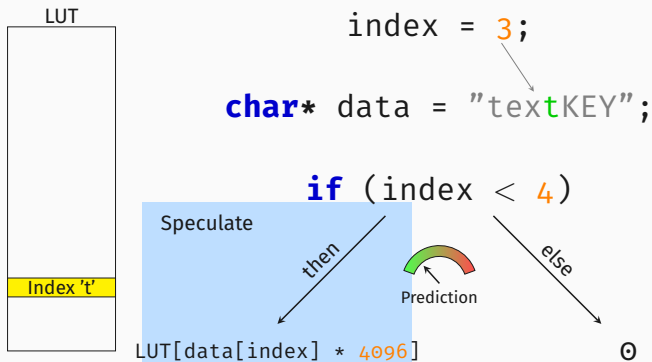


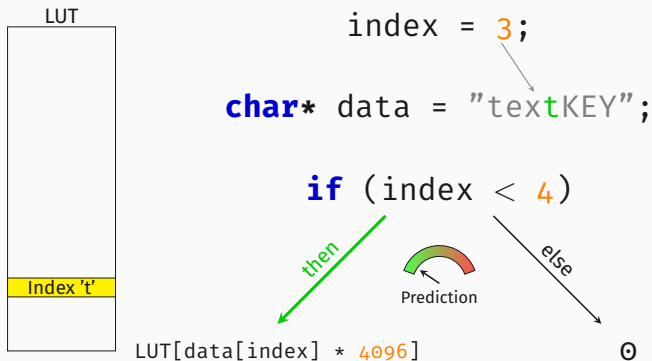


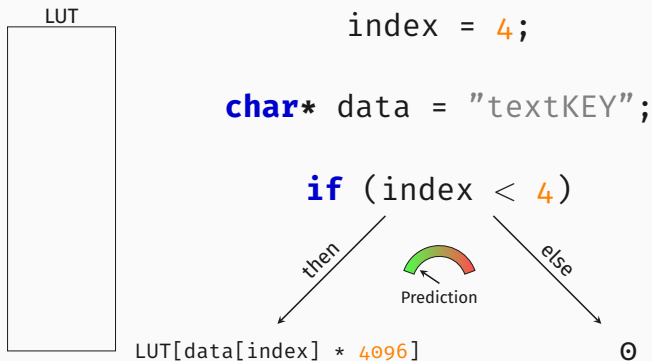


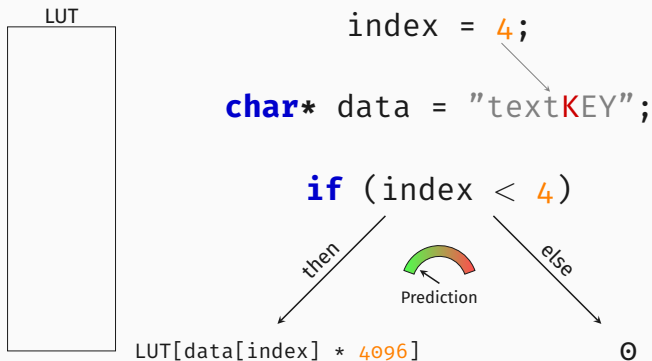


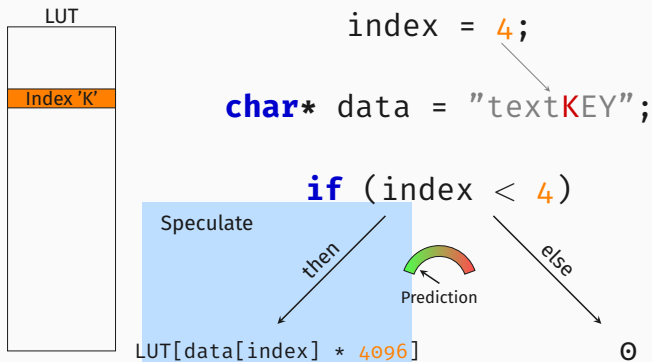


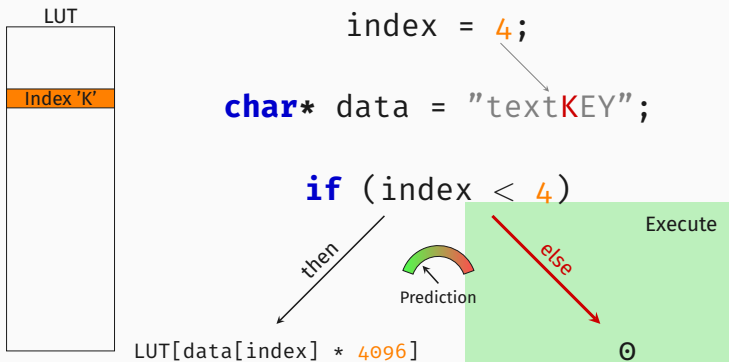


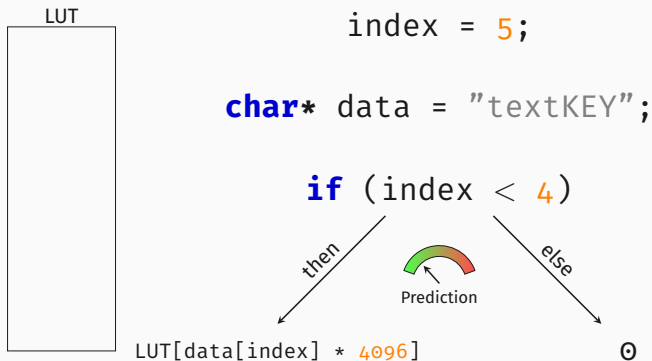


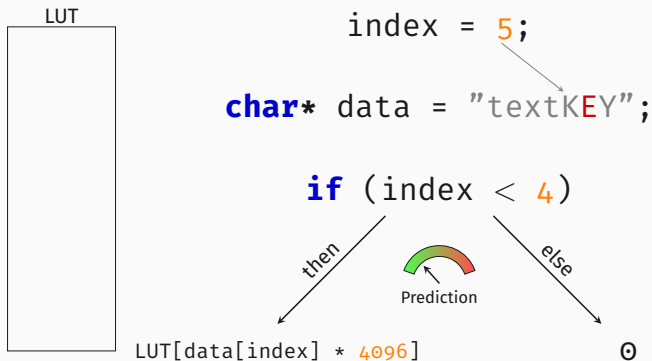


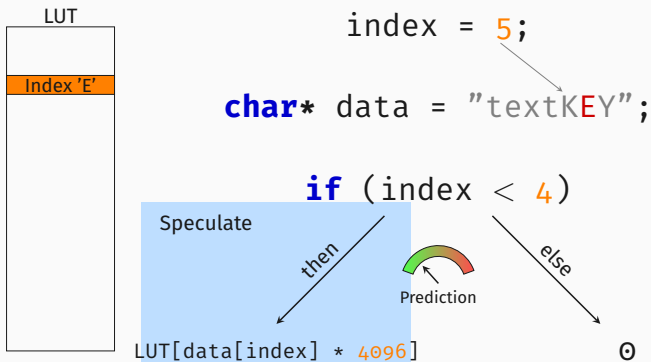


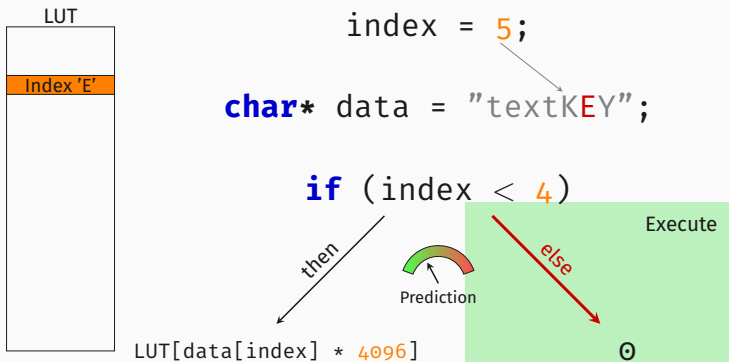


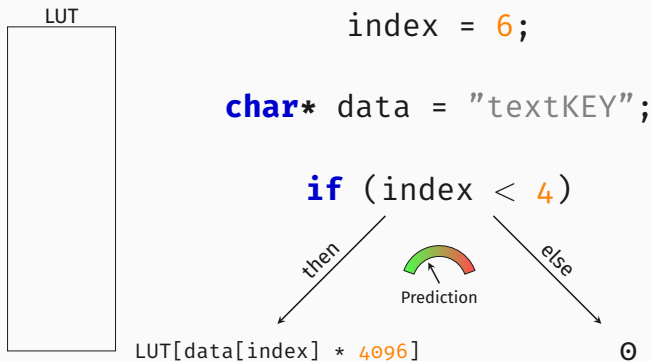


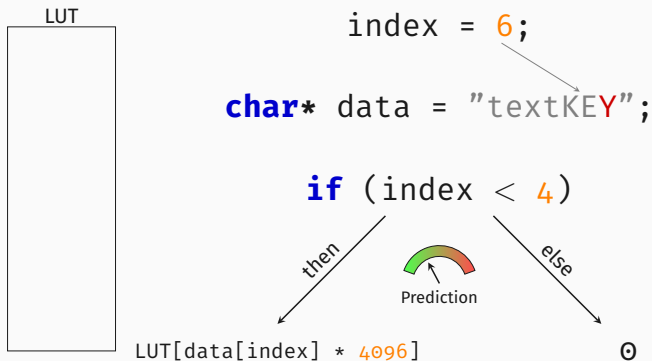


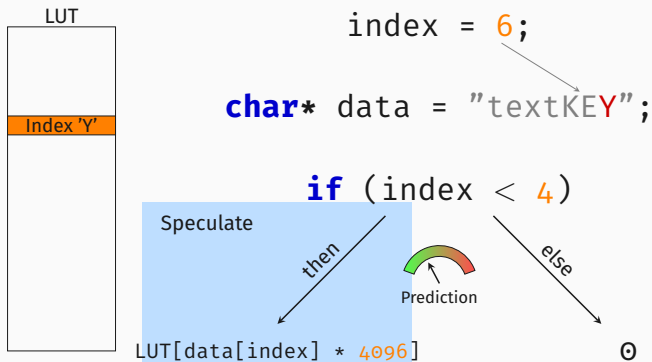


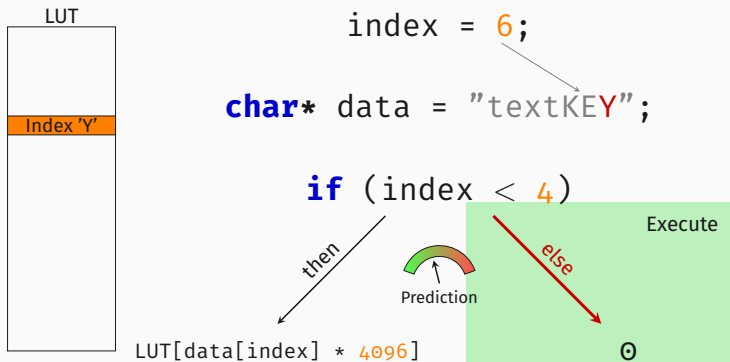


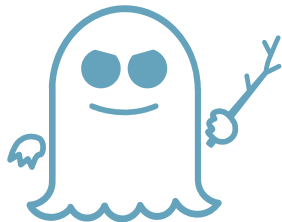
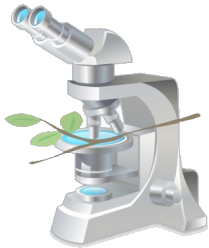












?

?

Meltdown: Transiently encoding unauthorized memory



Unauthorized access

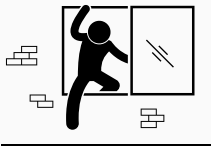
Listing 1: x86 assembly

```
1 meltdown:
2   // %rdi: oracle
3   // %rsi: secret_ptr
4
5   movb (%rsi), %al
6   shl $0xc, %rax
7   movq (%rdi, %rax), %rdi
8   retq
```

Listing 2: C code.

```
1 void meltdown(
2     uint8_t *oracle,
3     uint8_t *secret_ptr)
4 {
5     uint8_t v = *secret_ptr;
6     v = v * 0x1000;
7     uint64_t o = oracle[v];
8 }
```

Meltdown: Transiently encoding unauthorized memory



Unauthorized access



Transient out-of-order window

Listing 1: x86 assembly.

```
1 meltdown:
2   // %rdi: oracle
3   // %rsi: secret_ptr
4
5   movb (%rsi), %al
6   shl $0xc, %rax
7   movq (%rdi, %rax), %rdi
8   retq
```

Listing 2: C code.

```
1 void meltdown(
2     uint8_t *oracle,
3     uint8_t *secret_ptr)
4 {
5     uint8_t v = *secret_ptr;
6     v = v * 0x1000;
7     uint64_t o = oracle[v];
8 }
```

oracle array



secret idx

Meltdown: Transiently encoding unauthorized memory



Unauthorized access

Transient out-of-order window

Exception

(discard architectural state)

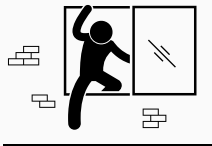
Listing 1: x86 assembly.

```
1 meltdown:  
2  // %rdi: oracle  
3  // %rsi: secret_ptr  
4  
5  movb (%rsi), %al  
6  shl $0xc, %rax  
7  movq (%rdi, %rax), %rdi  
8  retq
```

Listing 2: C code.

```
1 void meltdown(  
2     uint8_t *oracle,  
3     uint8_t *secret_ptr)  
4 {  
5     uint8_t v = *secret_ptr;  
6     v = v * 0x1000;  
7     uint64_t o = oracle[v];  
8 }
```

Meltdown: Transiently encoding unauthorized memory



Unauthorized access



Transient out-of-order window



Exception handler

Listing 1: x86 assembly.

```
1 meltdown:
2   // %rdi: oracle
3   // %rsi: secret_ptr
4
5   movb (%rsi), %al
6   shl $0xc, %rax
7   movq (%rdi, %rax), %rdi
8   retq
```

Listing 2: C code.

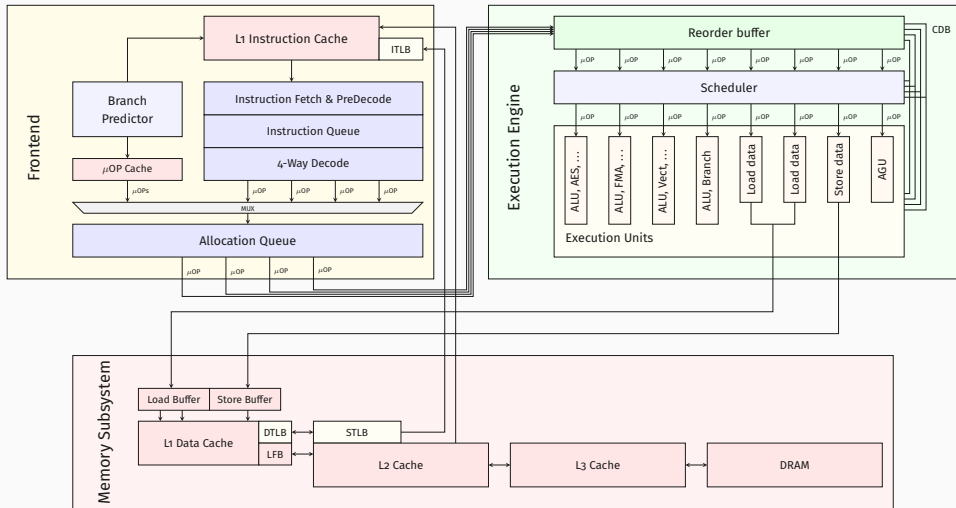
```
1 void meltdown(
2     uint8_t *oracle,
3     uint8_t *secret_ptr)
4 {
5     uint8_t v = *secret_ptr;
6     v = v * 0x1000;
7     uint64_t o = oracle[v];
8 }
```

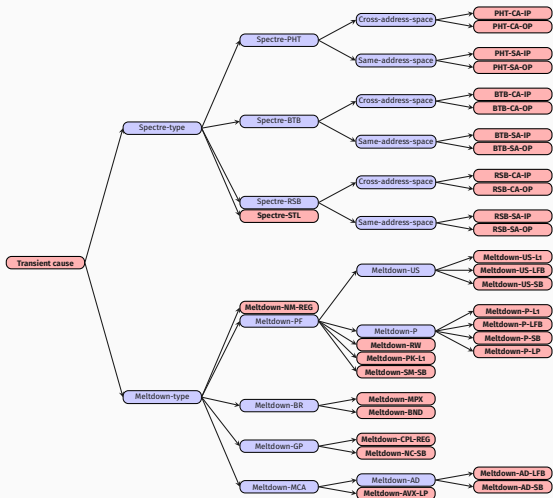
oracle array

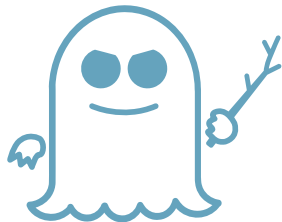
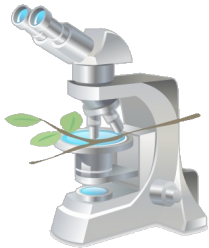


cache hit

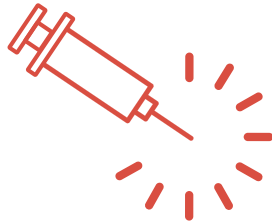
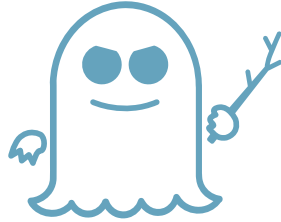
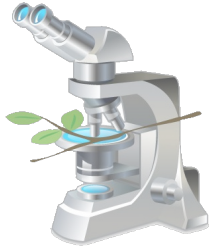
Meltdown variants: Microarchitectural buffers

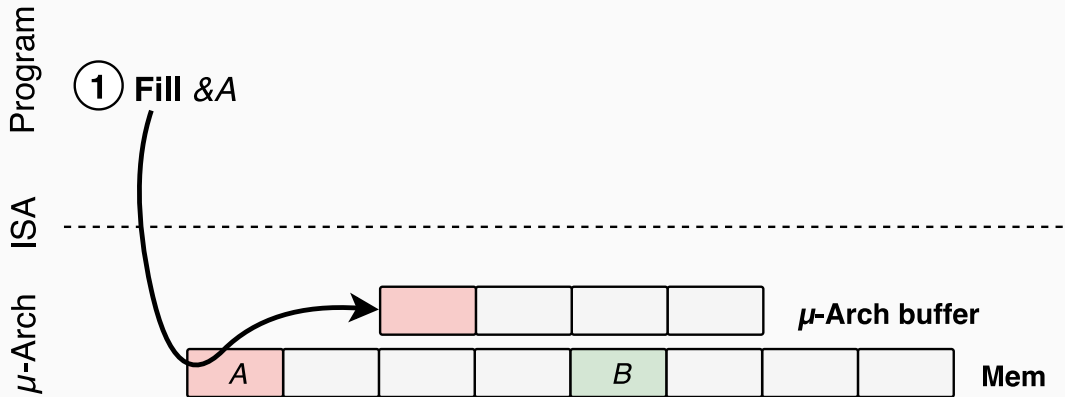


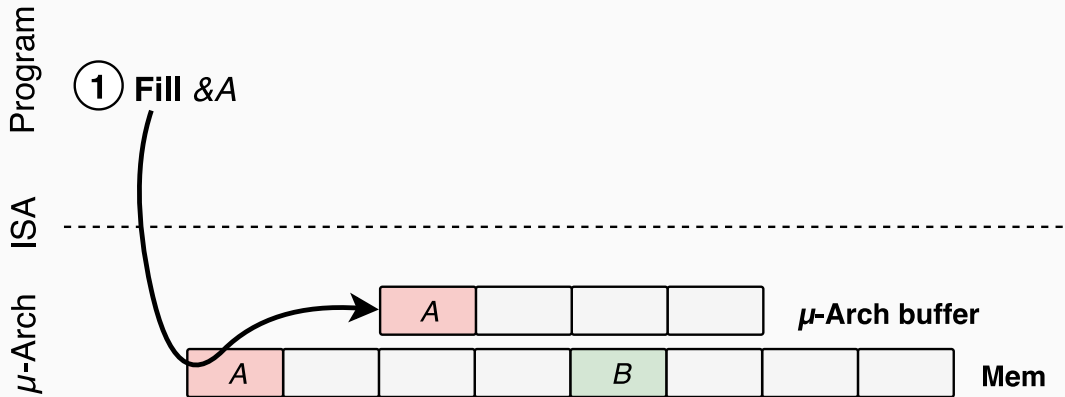


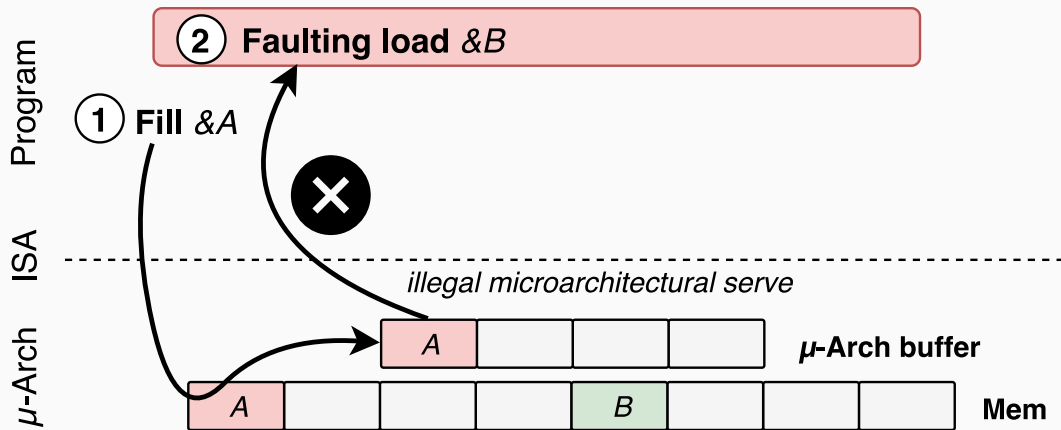


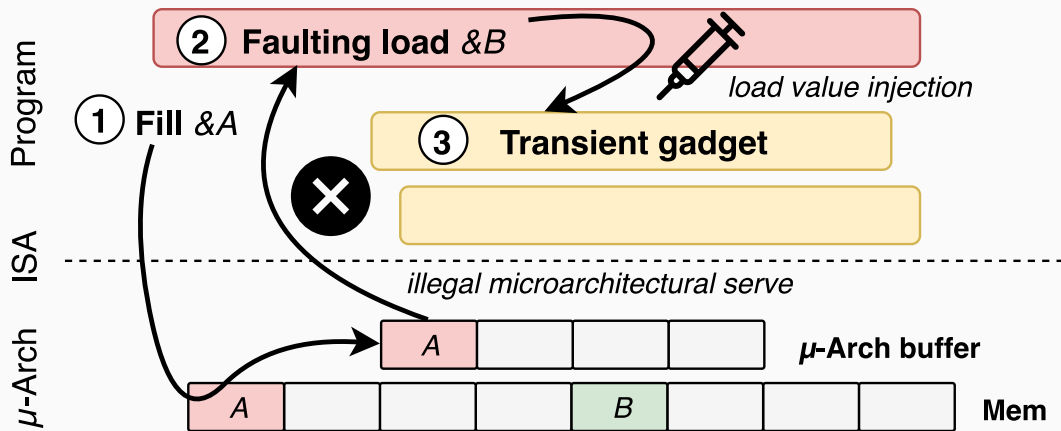
?

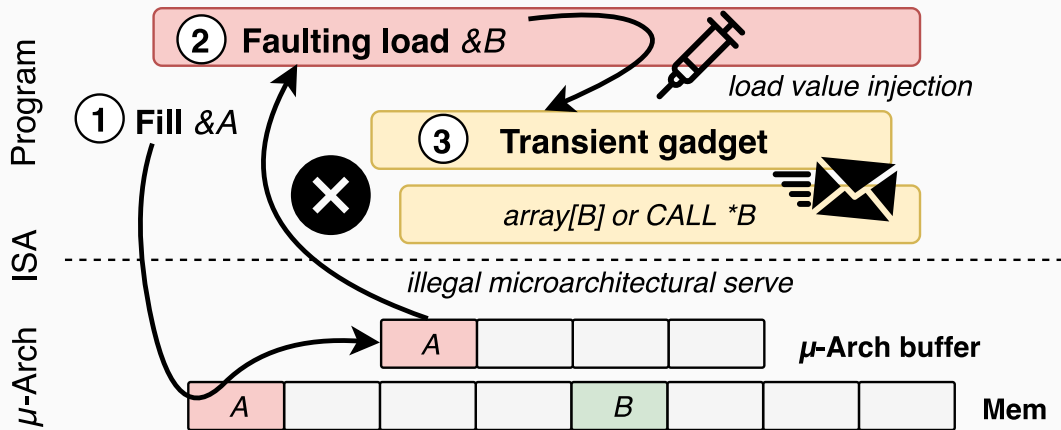


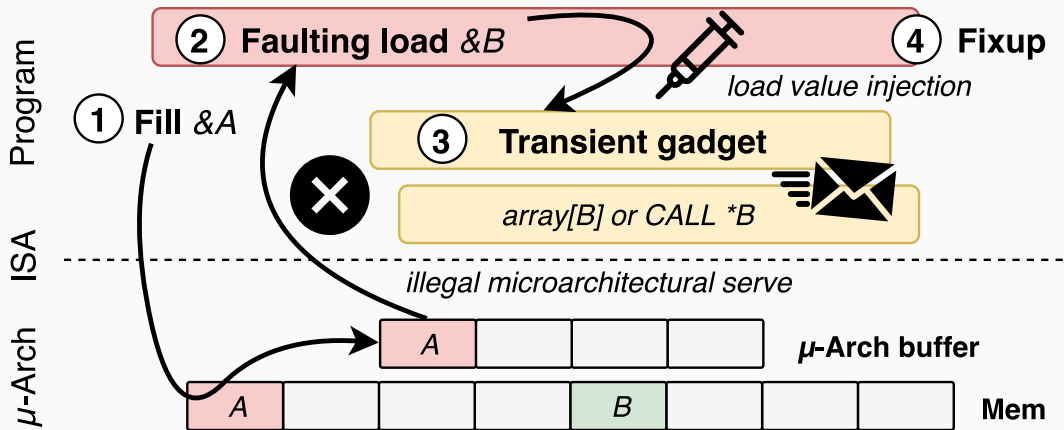








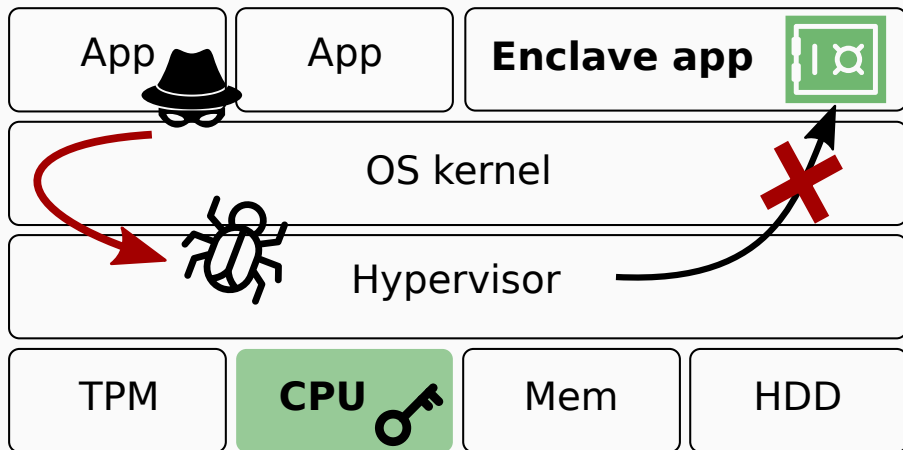




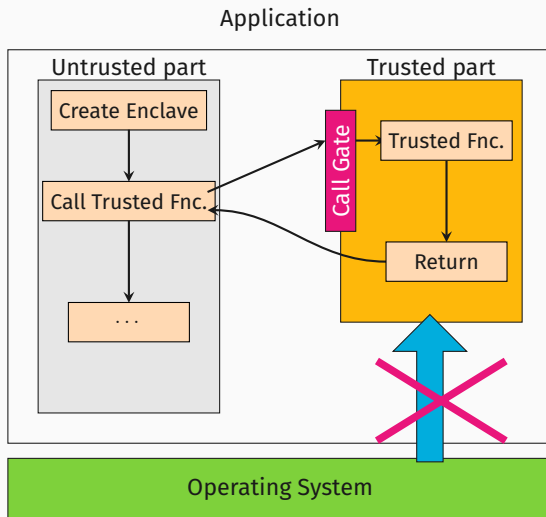
BUT WAIT...

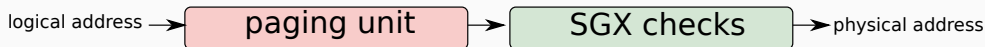
A PAGE FAULT IN THE VICTIM??

Enclaves to the rescue!

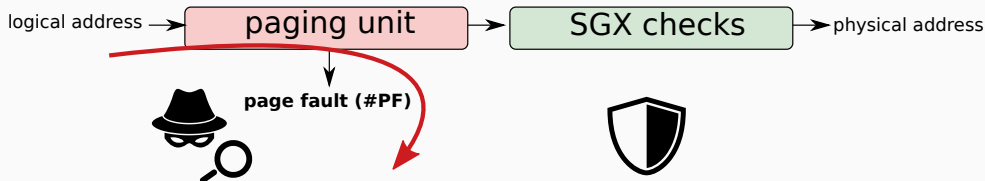


Intel SGX promise: hardware-level **isolation and attestation**

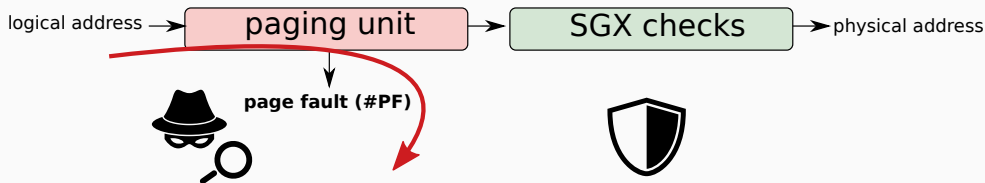




- **SGX machinery** protects against direct address remapping attacks



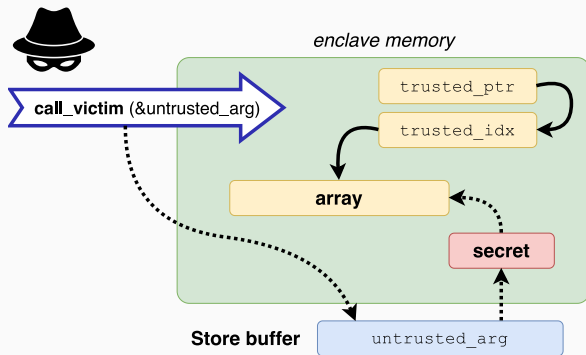
- **SGX machinery** protects against direct address remapping attacks
- ...but untrusted address translation may **fault** during enclaved execution (!)



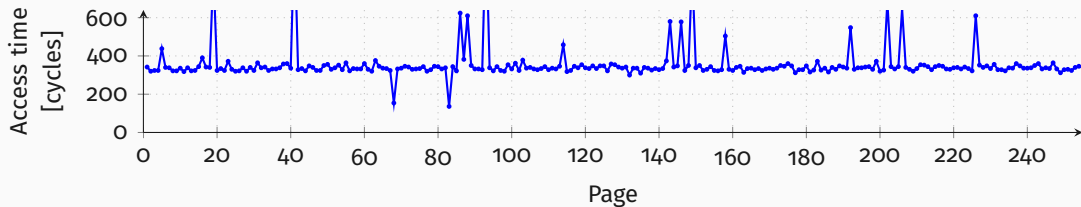
We can arbitrarily provoke page faults for trusted enclave loads!

A toy example

```
1 void call_victim(size_t
   untrusted_arg)
2 {
3     *arg_copy = untrusted_arg;
4     array[**trusted_ptr * 4096];
5 }
```



A Toy Example: Recovering arbitrary secrets



Real-world LVI ROP gadget in SGX-SDK

```
1 ; %rbx: user-controlled argument ptr (outside enclave)
2 sgx_my_sum_bridge:
3     ...
4     call my_sum          ; compute 0x10(%rbx) + 0x8(%rbx)
5     mov %rax, (%rbx)     ; P1: store sum to user address
6     xor %eax, %eax
7     pop %rbx
8     ret                 ; P2: load from trusted stack
9
```

Real-world LVI ROP gadget in SGX-SDK

```
1 ; %rbx: user-controlled argument ptr (outside enclave)
2 sgx_my_sum_bridge:
3     ...
4     call my_sum          ; compute 0x10(%rbx) + 0x8(%rbx)
5     mov %rax, (%rbx)     ; P1: store sum to user address
6     xor %eax, %eax
7     pop %rbx
8     ret                  ; P2: load from trusted stack
9
```



We can setup a fake transient stack in the store buffer!

Real-world LVI-ROP gadget in Quoting Enclave



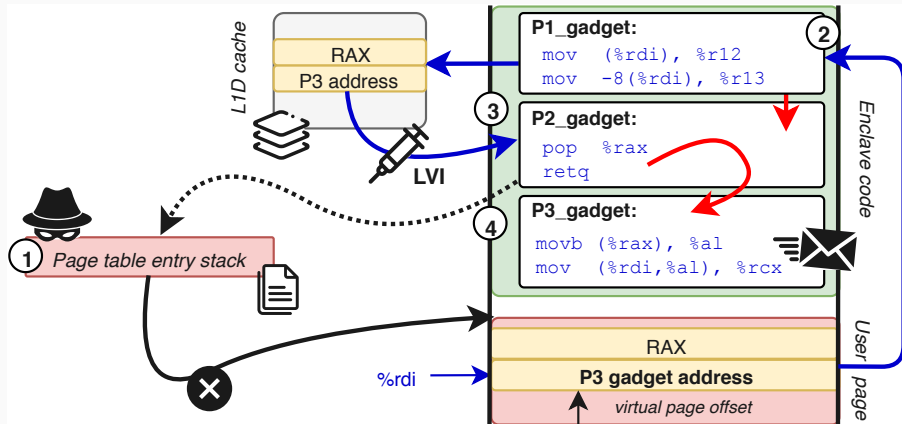
```
1 __intel_avx_rep_memcpy: ; libirc_2.4/efi2/libirc.a
2   ...                  ; P1: store to user address
3   vmovups %xmm0,-0x10(%rdi,%rcx,1)
4   ...
5   pop      %r12         ; P2: load from trusted stack
6   ret
7
```



```
danngdann-XP5-8920:~/Projects/lvl/lvl-rop-short$
```

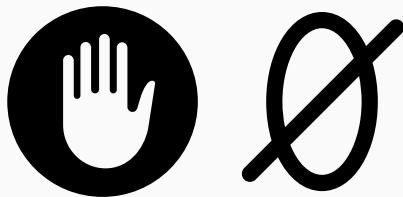


Worse: “Inverse Foreshadow” can remap the L1D cache!



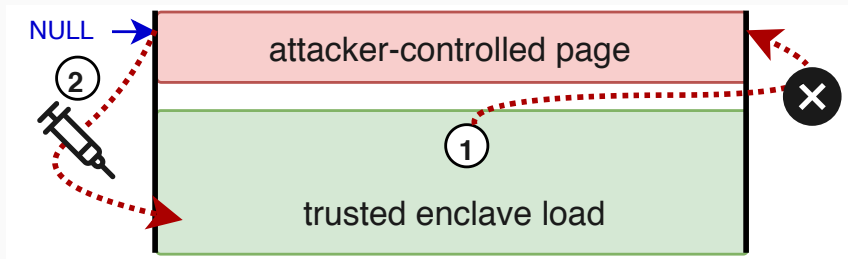
BUT ARE RECENT INTEL CPUS NOT

MELTDOWN-RESISTANT?



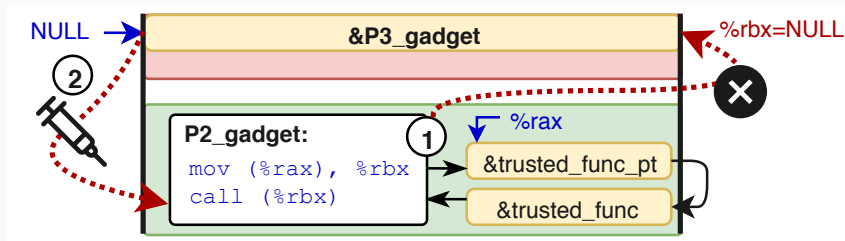
- Recent Intel CPUs forward **0x00 dummy values** for faulting loads

LVI-NULL: Why 0x00 is *not* a safe value



- Recent Intel CPUs forward **0x00 dummy values** for faulting loads
- ...but NULL is a **valid virtual memory address**, under attacker control

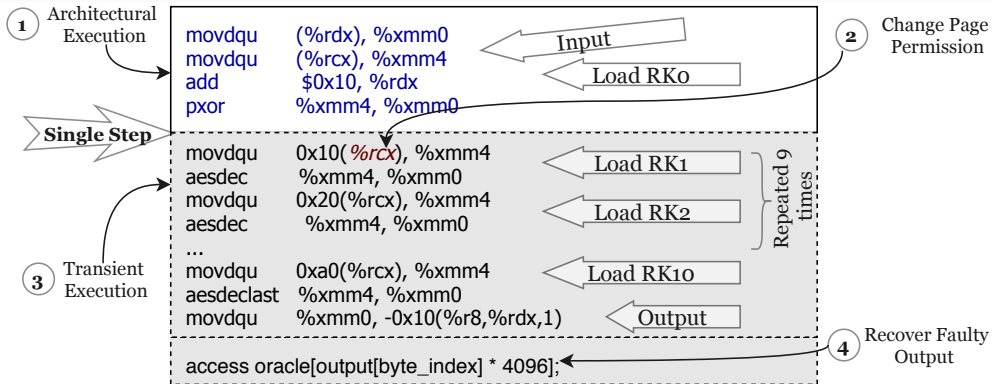
LVI-NULL: Why 0x00 is *not* a safe value



- Recent Intel CPUs forward **0x00 dummy values** for faulting loads
- ...but NULL is a **valid virtual memory address**, under attacker control
- ...hijack function pointer-to-pointer

```
1 asm_oret: ; (linux-sgx/sdk/trts/linux/trts_pic.S#L454)
2     ...
3     mov     ox58(%rsp),%rbp           ; %rbp ← NULL
4     ...
5     mov     %rbp,%rsp               ; %rsp ← NULL
6     pop     %rbp                   ; %rbp ← *(NULL)
7     ret                               ; %rip ← *(NULL+8)
8
```

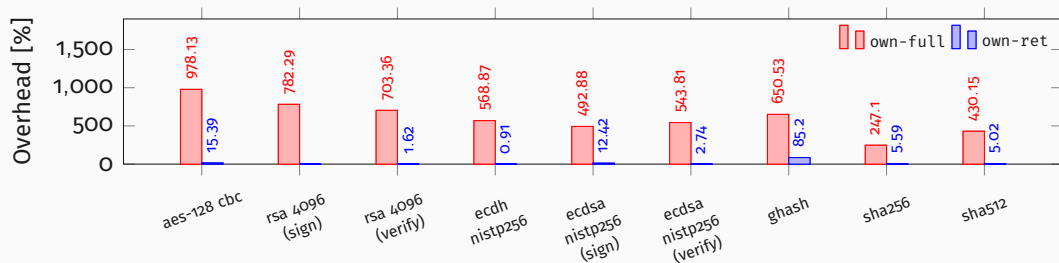
LVI-NULL fault injection to reduce AES rounds



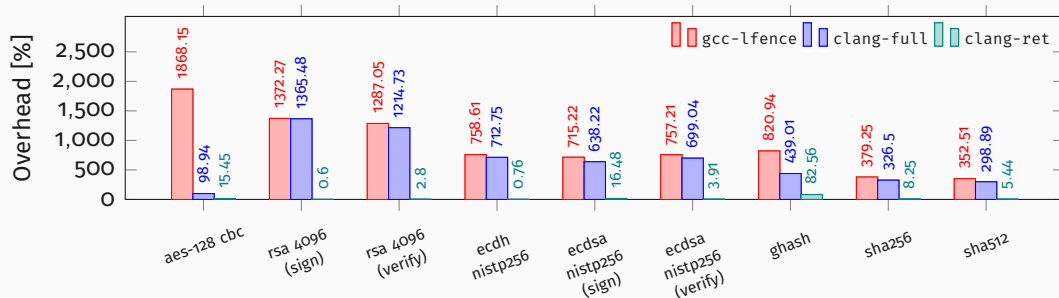
danm@danm-XPS-8920:~/Projects/lvl/sgx-aesni-zero\$

Instruction	Possible Emulation	Clobber
ret	pop %reg; lfence; jmp *%reg	✓
ret	not (%rsp); not (%rsp); lfence; ret	✗
jmp (mem)	mov (mem),%reg; lfence; jmp *%reg	✓
call (mem)	mov (mem),%reg; lfence; call *%reg	✓

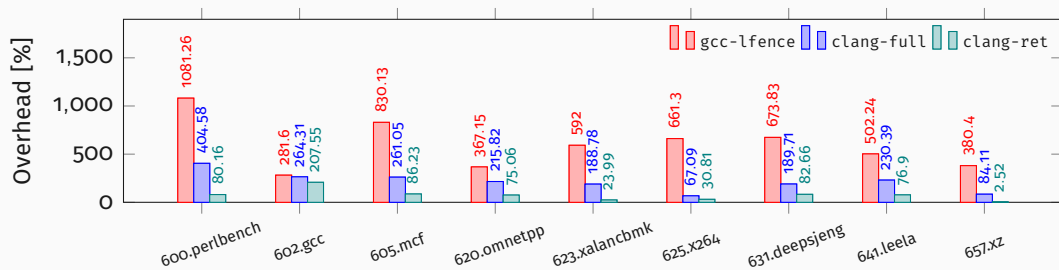
Performance Overheads (Our Mitigation)



Performance Overheads (Intel's Mitigation)



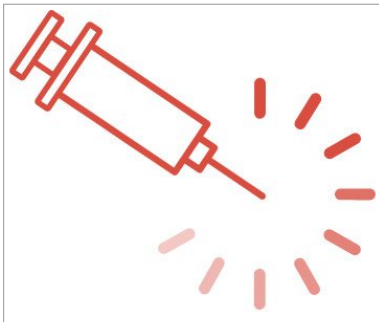
Performance Overheads (Intel's Mitigation)



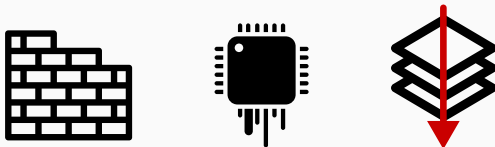
The Brutal Performance Impact From Mitigating The LVI Vulnerability

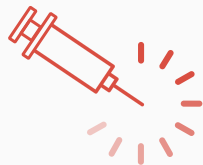
Written by [Michael Larabel](#) in [Software](#) on 12 March 2020. **Page 1 of 6.** [76 Comments](#)

On Tuesday [the Load Value Injection \(LVI\) attack was disclosed](#) by Intel and security researchers as a new class of transient-execution attacks and could lead to injecting data into a victim program and in turn stealing data, including from within SGX enclaves. While Intel has publicly stated they don't believe the LVI attack to be practical, one of their open-source compiler wizards did go ahead and add [mitigation options to the GNU Assembler](#) as part of the GCC toolchain. Here are benchmarks showing the performance impact of enabling those new LVI mitigation options and the significant impact they can cause on run-time performance in real-world workloads.



- ⇒ New emerging and powerful class of **transient-execution** attacks
- ⇒ Importance of fundamental **side-channel** research
- ⇒ Security **cross-cuts** the system stack: hardware, hypervisor, kernel, compiler, application





LVI

Hijacking Transient Execution with Load Value Injection

Daniel Gruss, Daniel Moghimi, Jo Van Bulck

Hardwear.io Virtual Con, April 30, 2020