中国科学院大学
University of Chinese Academy of Sciences

论文Review: LVI: Hijacking Transient Execution through Microarchitectural Load Value Injection (S&P 20') CCF-A

Jo Van Bulck∗, Daniel Moghimi†, Michael Schwarz‡, Moritz Lipp‡, Marina Minkin §, Daniel Genkin §, Yuval Yarom¶, Berk Sunar†, Daniel Gruss‡, and Frank Piessens∗

∗ DistriNet KU LEUVEN

† Worcester Polytechnic Institute

‡ TU Graz Graz University of Technology

§ The University of Michigan

¶ The University of Adelaide, CSIRO, Data61

Jing Li

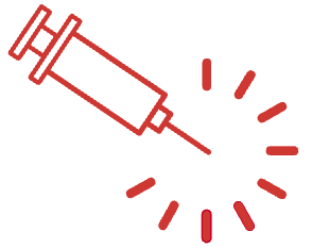
Apr. 28, 2021

辛丑三月十七

Huairou, Beijing

中国科学院信息工程研究所
INSTITUTE OF INFORMATION ENGINEERING,CAS

Executive summary



Background：

- **现有CPU攻击，Meltdown等不够心机，有效率不够**
  - ▶ 分支预测有侧信道
  - ▶ 攻击模式可结合，以绕过现有缓解方案

Key Idea：

- **逆向思维，高维打击。配合精心构建攻击方案实现由内而外的核打击**
- **与其将数据直接从受害者身上泄漏到攻击者，不如通过隐藏的处理器缓冲区将不正确的数据注入到受害者，再劫持瞬态执行**

Mechanisms ：

- **四步攻击：注入、读取、执行、hack，具体实现多种多样**

Results ：

- **大规模型号CPU受影响，从Intel SGX窃取敏感数据和密钥**
- **硬件解决，成本极高；软件临时缓解，计算速度降低2-19倍**
- **新型的攻击思路，但实际利用时难度极高**

Overview

Background, Problem & Goal

Recap: CPU Microarchitecture，微指令(micro-ops, μ-ops)

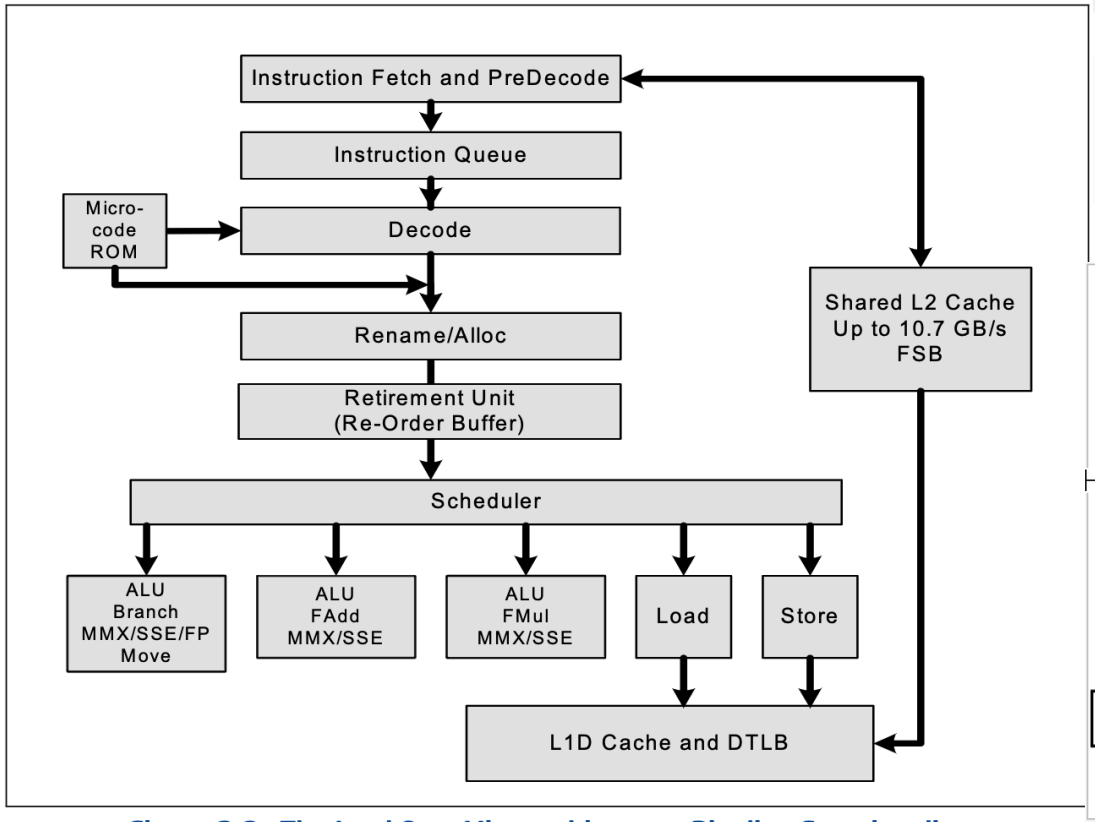
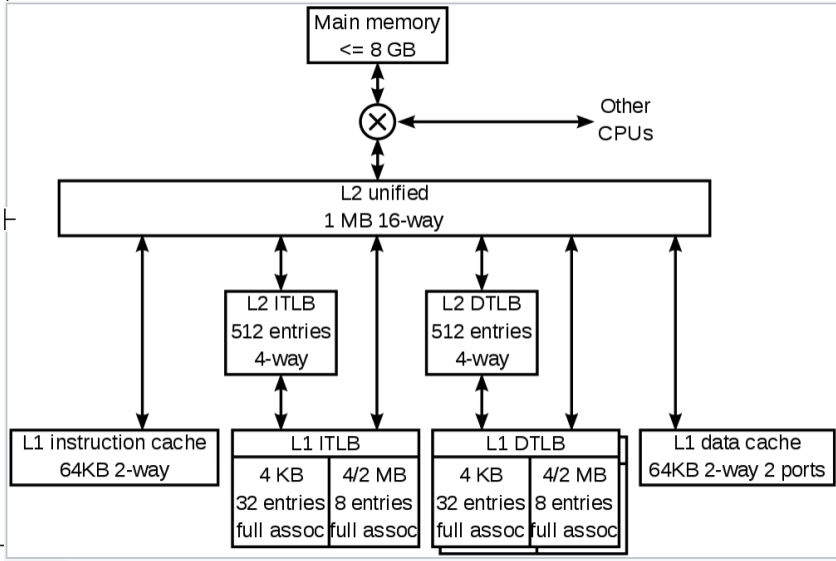


Figure 2-3. The Intel Core Microarchitecture Pipeline Functionality

```
pop [ebx]
```

⇨

```
load temp, [esp]
store [ebx], temp
add esp, 4
```

Cache hierarchy of the K8 core in the AMD Athlon 64 CPU.

Source: https://software.intel.com/content/www/us/en/develop/articles/intel-sdm.html

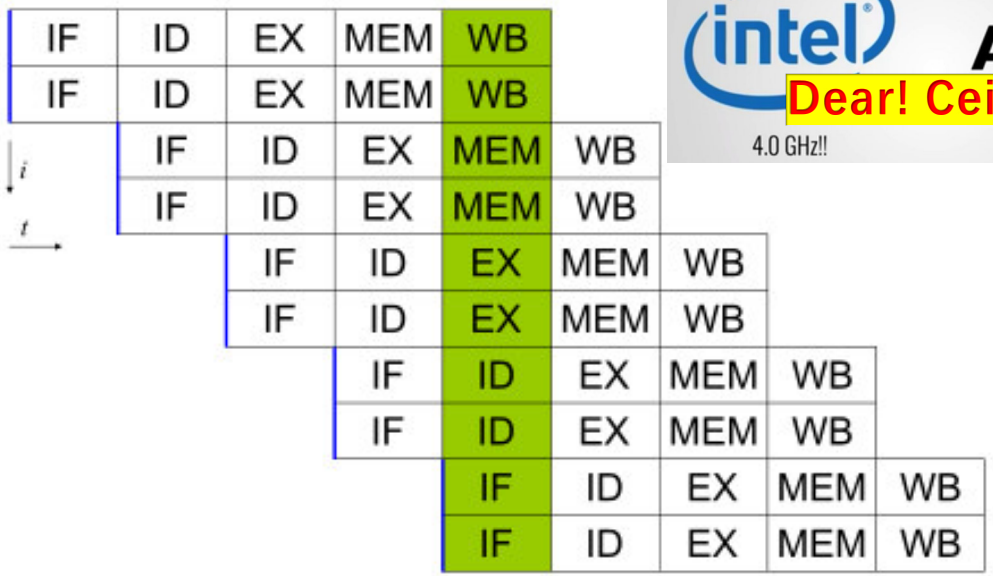
P	RW	US	WT	UC	A	D	S	G	Physical Page Number	Rsvd.	XD

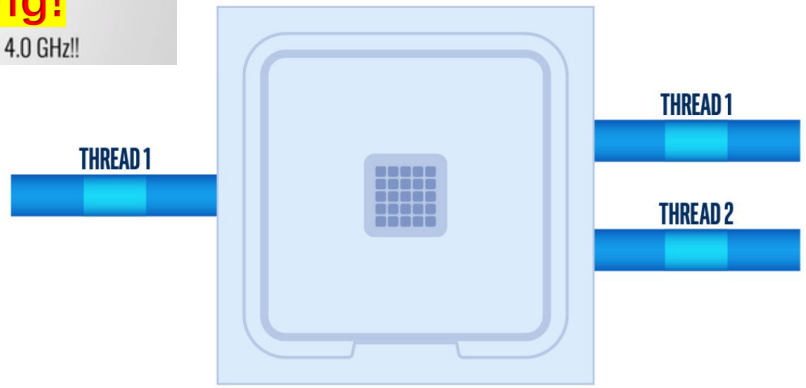
Fig. 1. Overview of an x86 page-table entry and attributes that may trigger architectural page fault exceptions (red bold) or microcode assists (green italic). Attributes that are periodically cleared by some OS kernels are underlined; all other fields can only be modified by privileged attackers.

Recap: Modern Microprocessors, Concurrent execution

- Pipeline
- Superpipeline
- Superscalar, 超标量 [CDC 6600, 1964]
- VLIW
- Hyper-Threading, 超线程 [Intel Xeon, 2002]

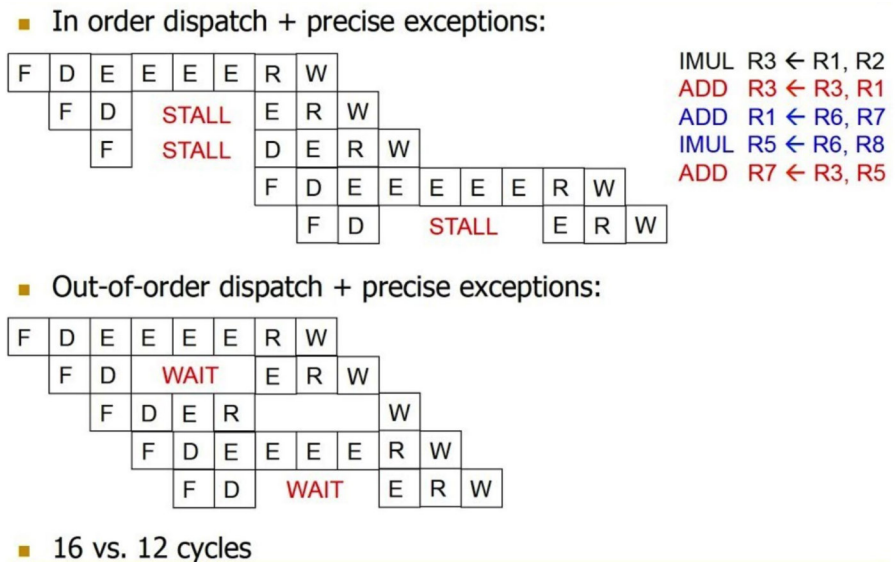
虚拟画饼，压榨CPU，007打工人

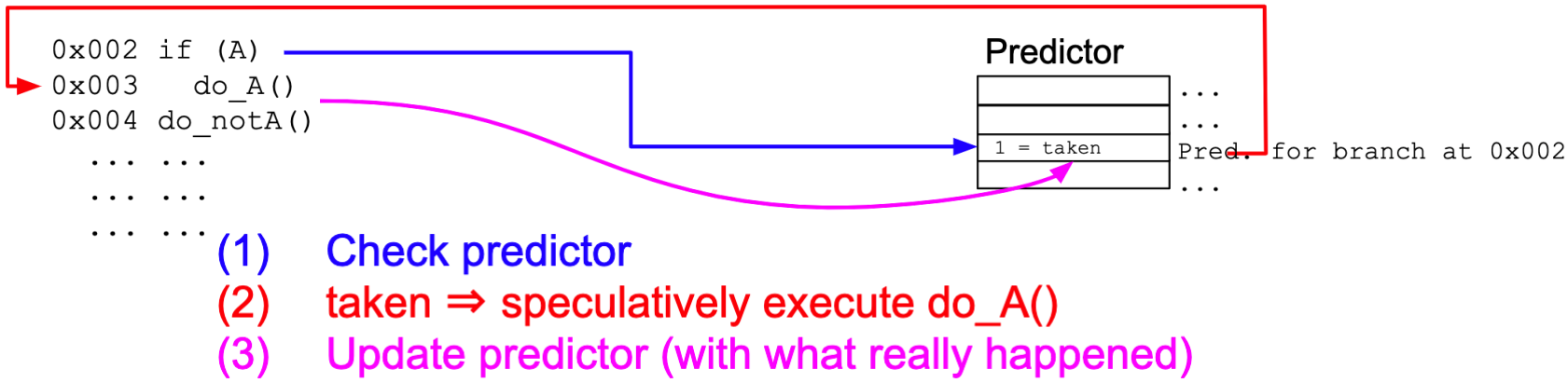




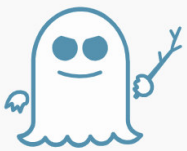
Recap: Out-of-Order Execution, avoid pipeline stalls

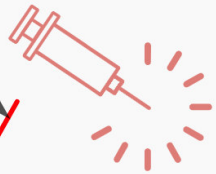
- Speculative Execution
- Multiple Branch Prediction
- data flow analysis
- Tomasulo (CPU) [IBM 360, 1967]
- Scoreboarding(GPU) [CDC 6600, 1964]



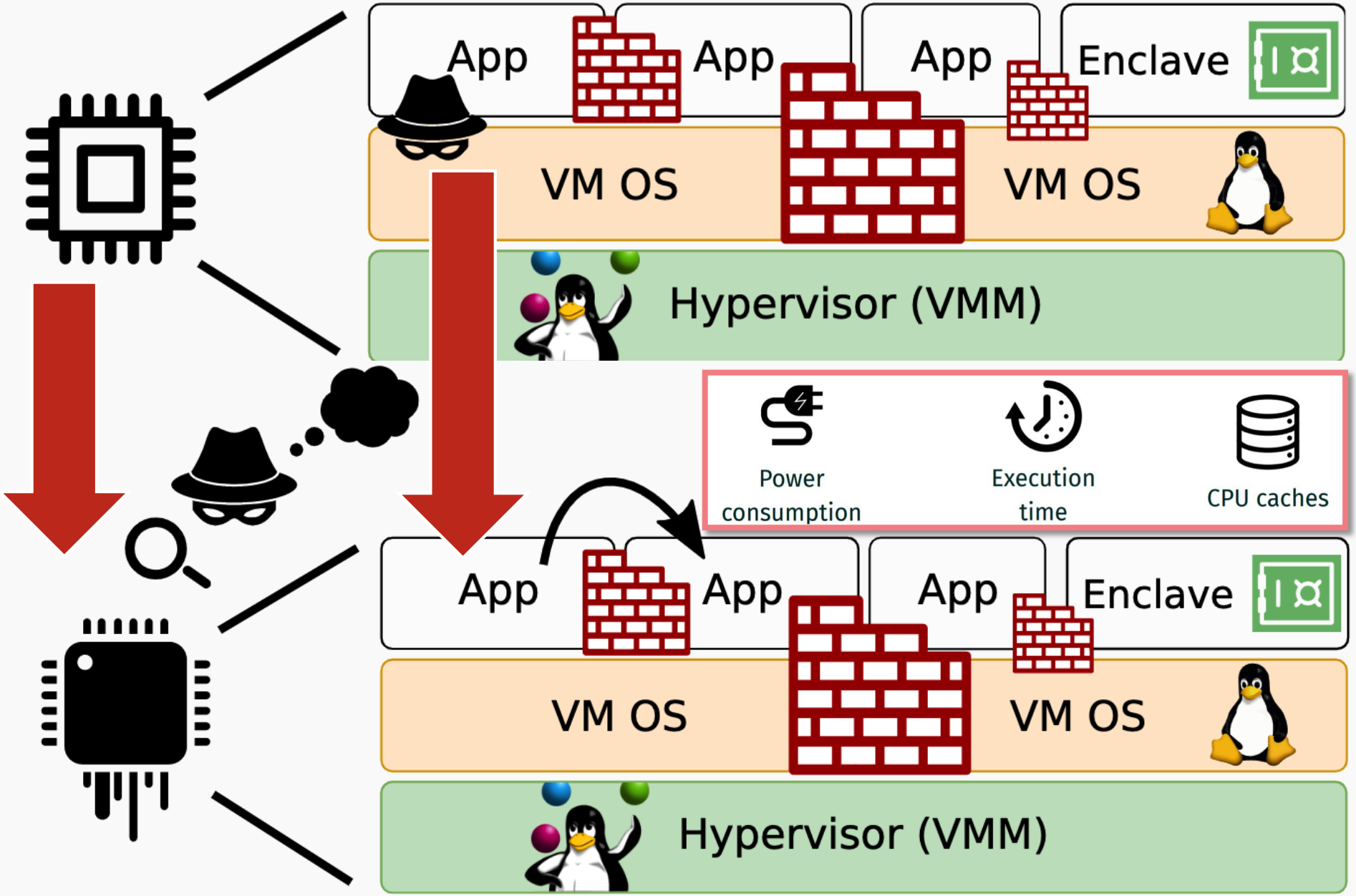


Side-channel attacks 20年前就有 What's new? (纵轴 occurrences in Google Scholar)

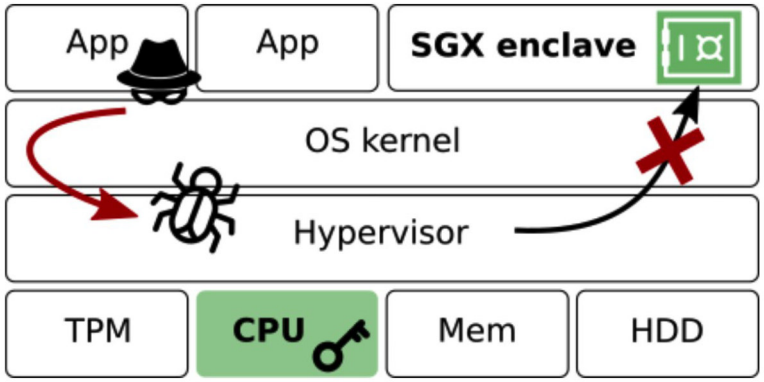




Recap: CPU (even SGX) remain leaky via Side-Channel Attack (Bypass)



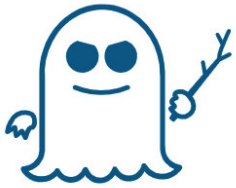
攻防博弈：SGX来了 攻击也来了



Intel SGX promise:
hardware-level isolation and attestation

Source: https://en.wikipedia.org/wiki/Software_Guard_Extensions

Microarchitecture Side-channel： Transient Execution Attacks



Spectre

v1, v2, v4, v5,
Spectre-BTB,
Spectre-RSB,
ret2spec,
SGXPectre,
SmotherSpectre,
NetSpectre?

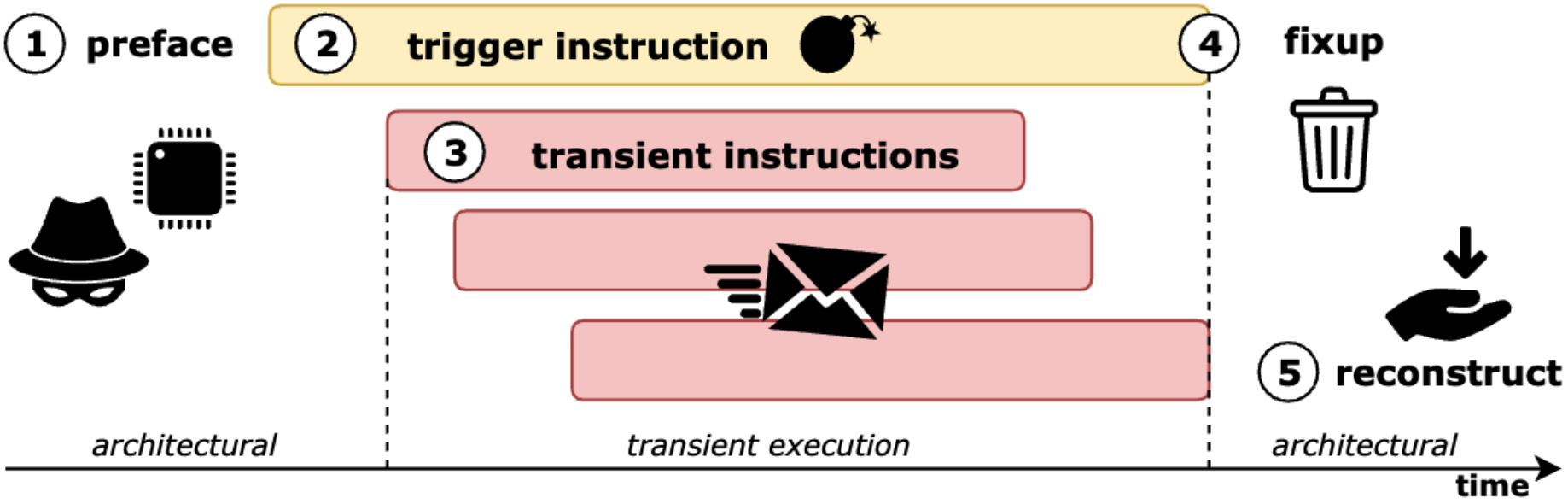
Meltdown

v3, v3.1, v3a,
RDCL?

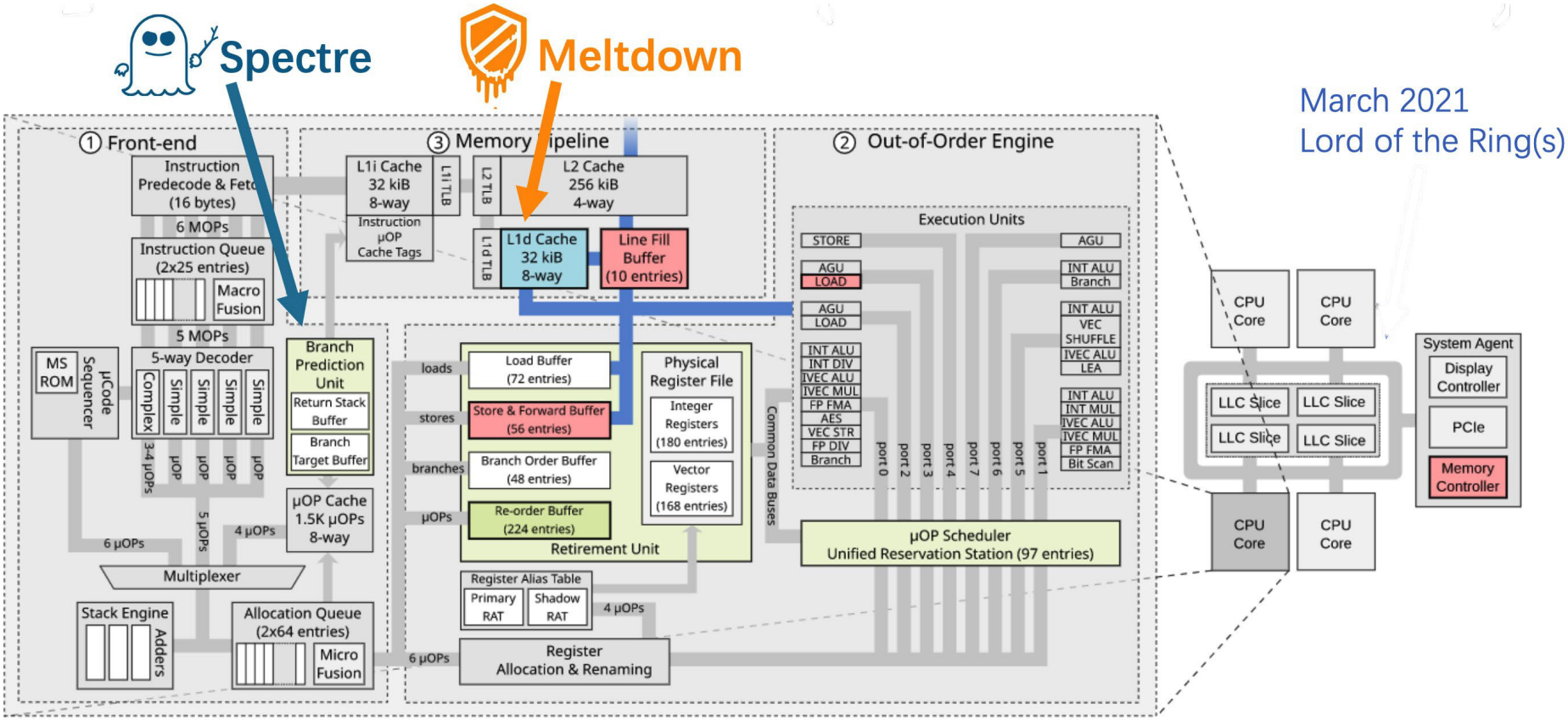
Foreshadow
Foreshadow-NG,
L1TF?

ZombieLoad, MDS?

RIDL, Fallout?

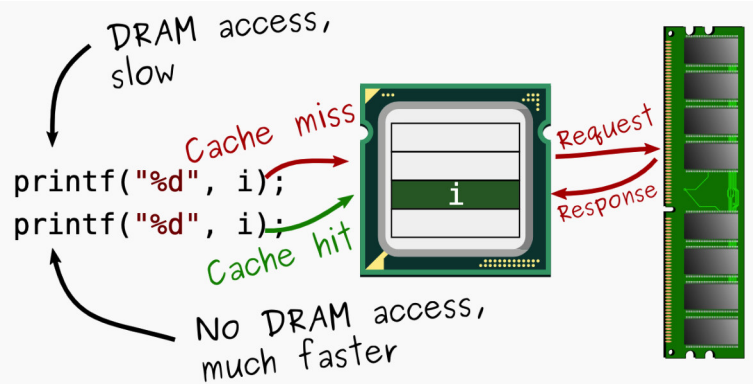


Skylake Microarchitecture (hacked continuously)

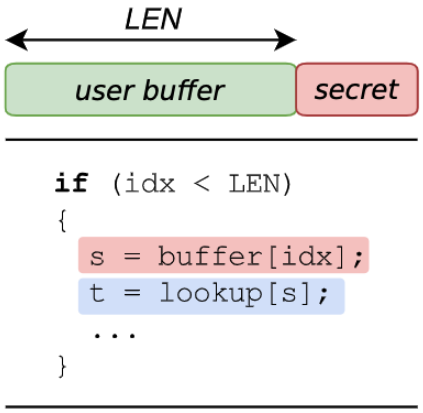


Source: https://mdsattacks.com/diagram.html

Recap: cache-base attack to Spectre-PHT (v1)



类型	被缓存在何处	延迟（周期数）
CPU寄存器	芯片上的CPU寄存器	0
TLB	芯片上的TLB	0
L1 Cache	芯片上的一级缓存	4
L2 Cache	芯片上的二级缓存	10
L3 Cache	芯片上的三级缓存	50
虚拟内存	内存条	200

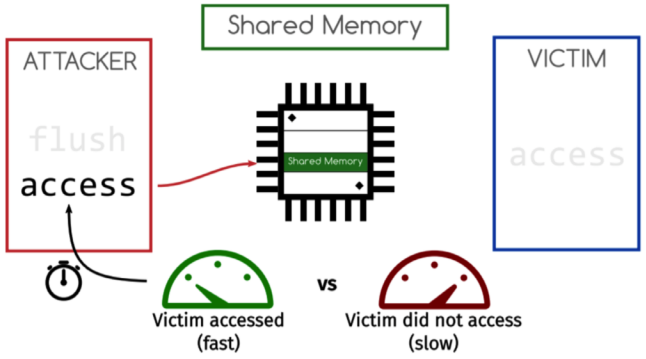


- Programmer *intention:* no out-of-bounds accesses
- **Mistrain gadget** to speculatively "ahead of time" execute with $idx \geq LEN$ in the transient world
- **Side channels** may leave traces after roll-back!

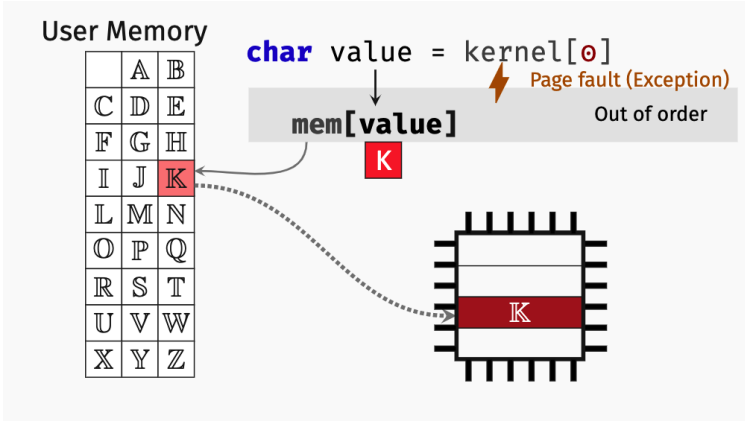
Spectre take-away

- CPU **transiently** executes wrong code paths
- **Confused-deputy gadgets** encode secrets via side channels

Recap: Flush + Reload to Meltdown



- It's possible to explicitly flush a cache line
  - `CLFLUSH`
- Same for measure the access latency
  - `RDTSC / RDTSCP`
- Non-privileged operations



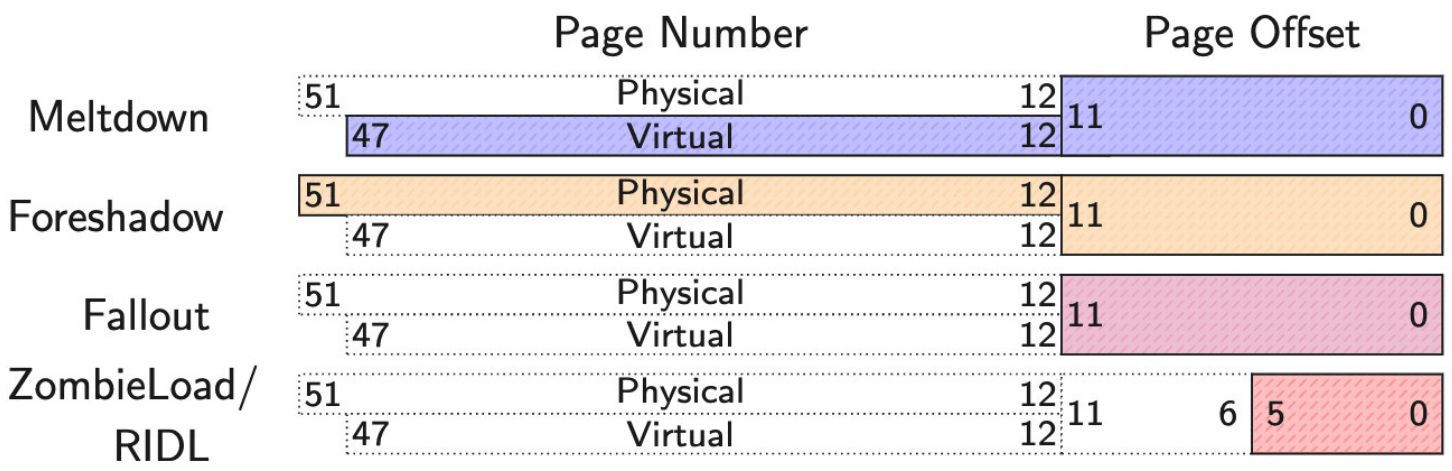
- Attacker先通过Flush清空对应的cache line
- 触发Victim访问该数据
- Attacker会访问同一数据并测量访问时间

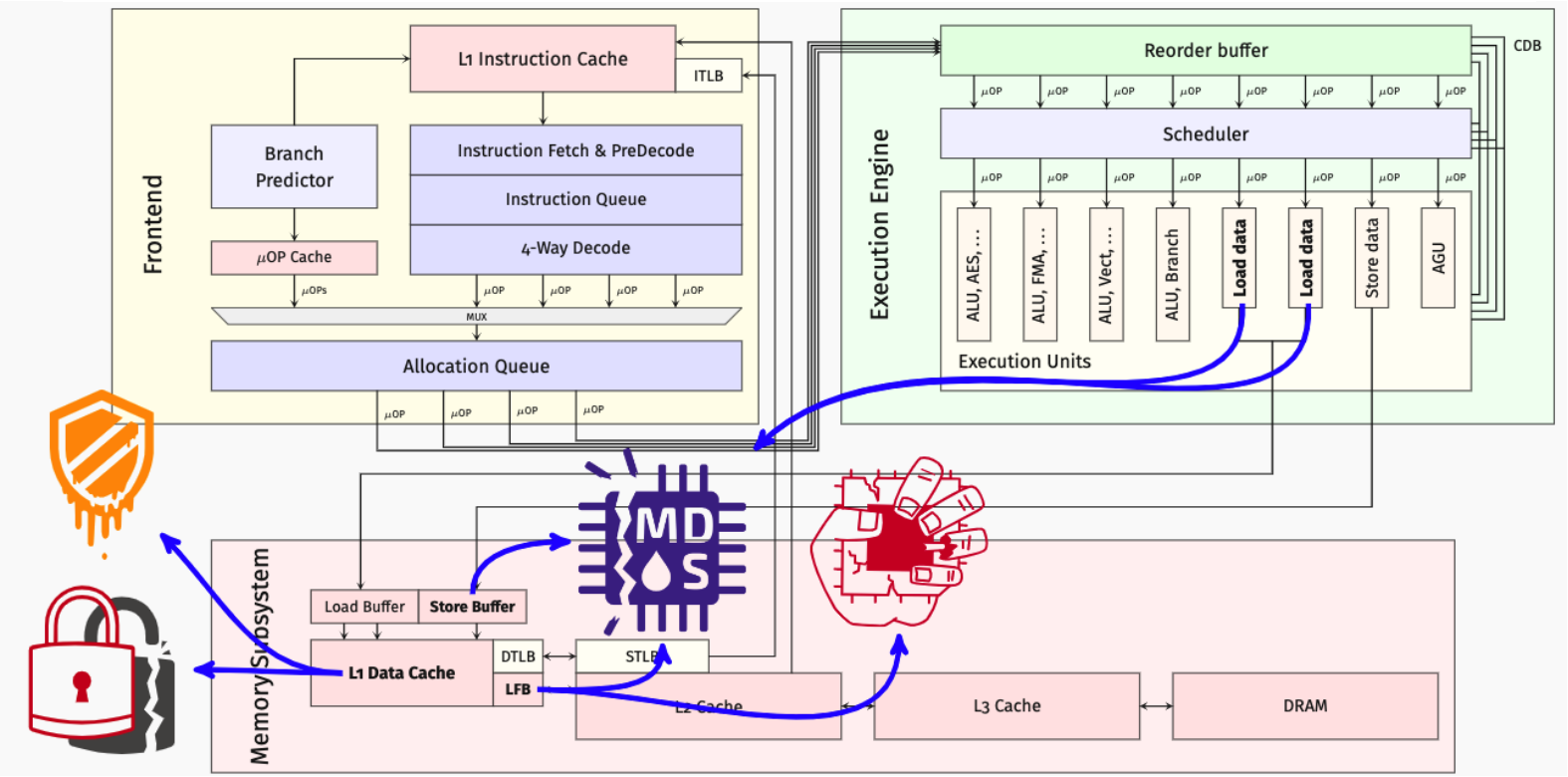
- CPU会在权限检查之前以乱序方式使用数据
- Meltdown可读取任何内核地址
- 物理内存通常映射在内核中→读取任意内存

Meltdown take-away

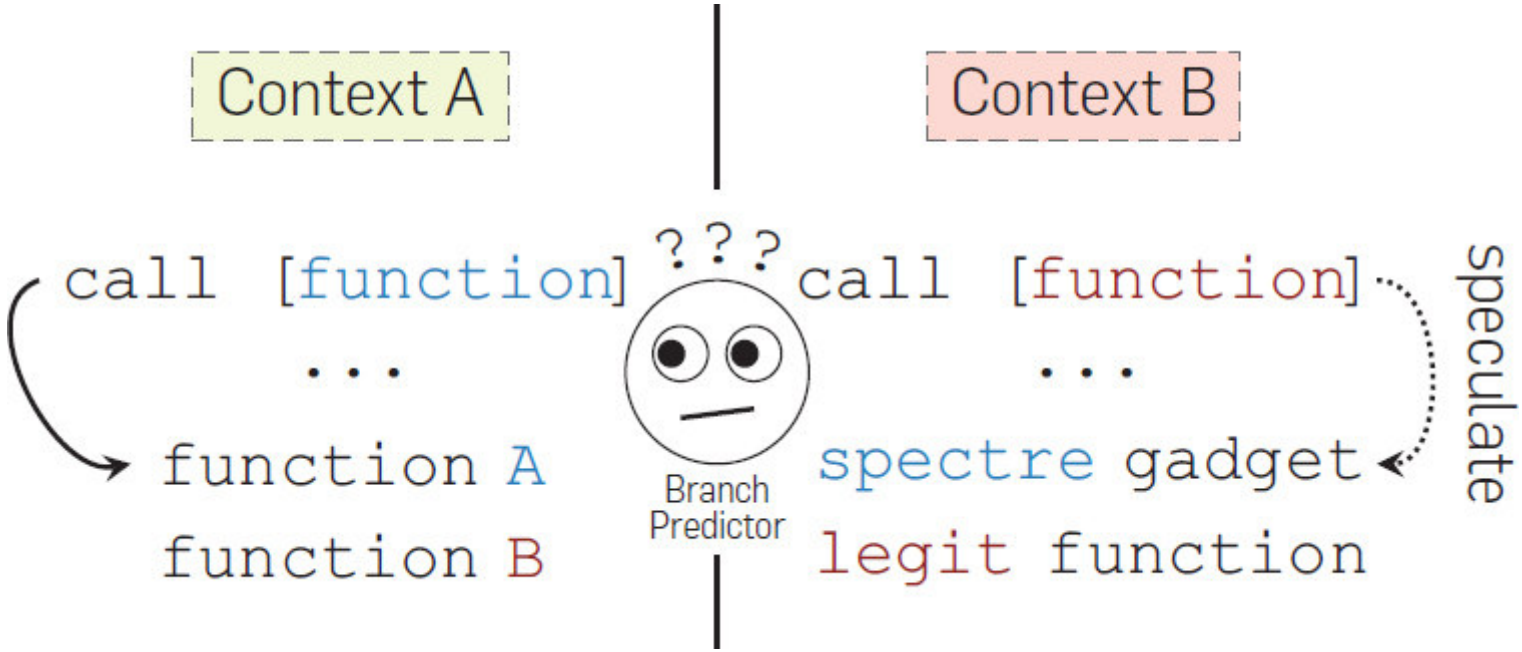
Faulting (or assisted) loads transiently forward **unrelated data** from various microarchitectural buffers

Meltdown variants: Microarchitectural buffers



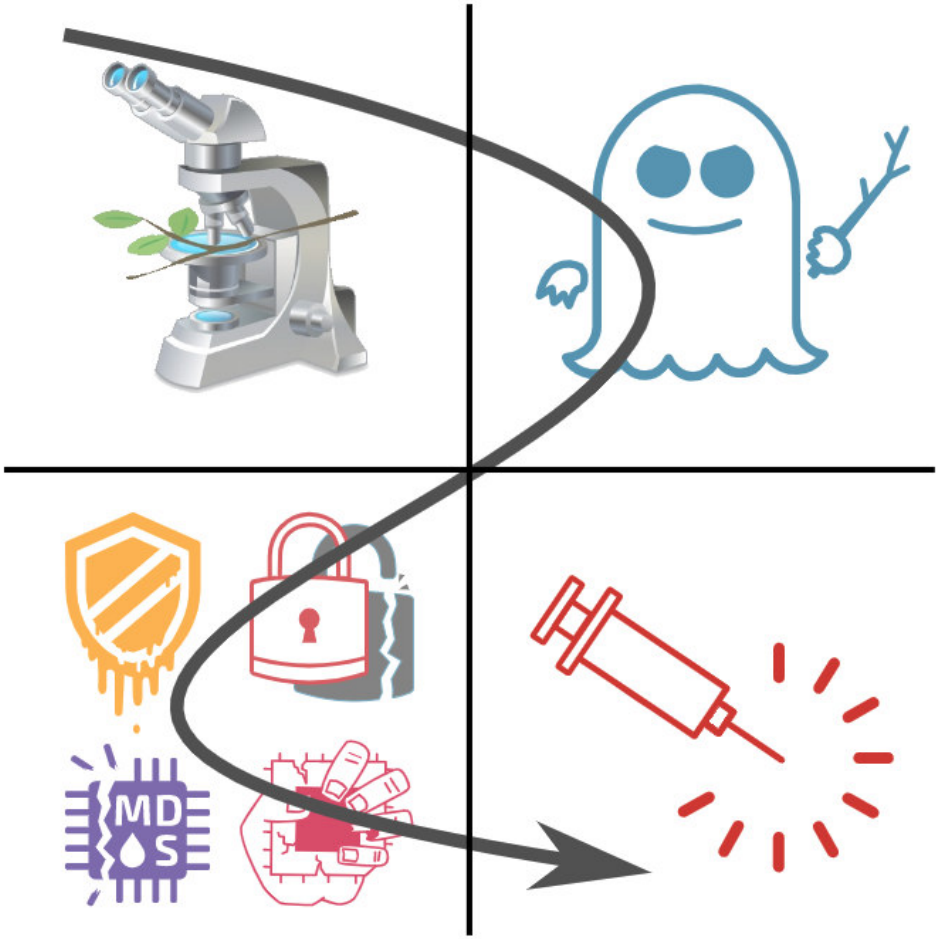


Spectre-v2：Poisoning Indirect Branches (CVE-2017-5715)



- 攻击者训练受害者（victim）进程所在的CPU的分支目标预测器（Branch Target Predictor）指向特定的代码片段（gadget），使得victim在特定间接跳转处预测失误（mispridict）进入到gadget。
- 在随后的处理器推测执行中执行gadget并将需要探测的信息映射到CPU的某一片缓存区域，最后攻击者通过逐一测量缓存区域内缓存块（Cache Line）命中时间得到命中位置，进而还原出所探测的信息。
- 核心问题在于CPU被攻击者误导转不过弯导致正常运行的进程内部的信息暴露给了攻击者。

Where LVI?



LVI = reverse Meltdown-like + Spectre v2-like-gadget

注入攻击者控制的数据转化为受害者的瞬态执行过程以利用之

Goal

逆向思维Meltdown实现注入式攻击，造成CPU核心数据泄露的可能性

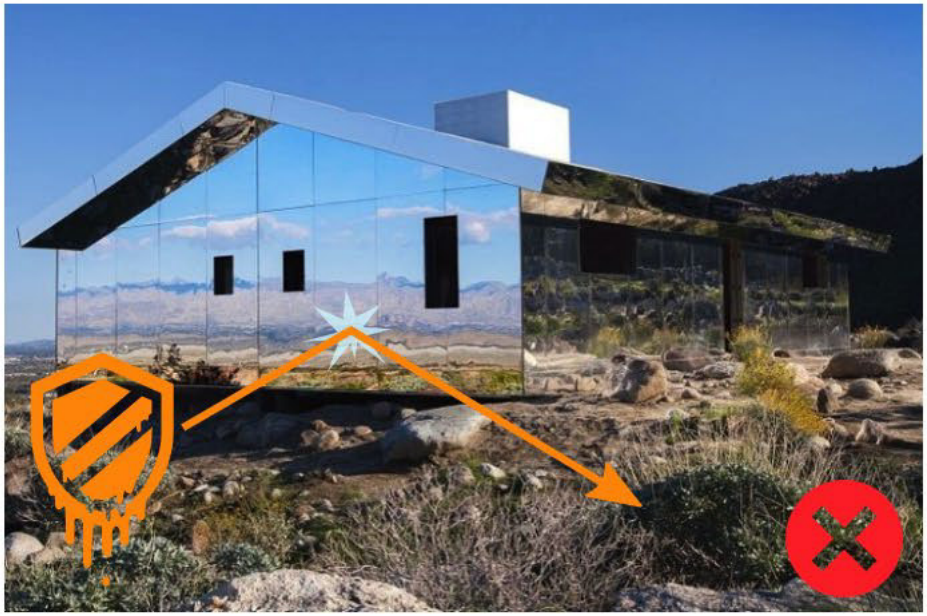
将数据注入到一个程序中，目的是替换从内存加载的值，然后在发现错误并回滚之前使用一小段时间，在此期间控制数据和控制流

LVI = reverse Meltdown-like + Spectre v2-like-gadget

注入攻击者控制的数据转化为受害者的瞬态执行过程以利用之

Key Approach & Ideas

Inception：意识注入 逆向思维



Outside view

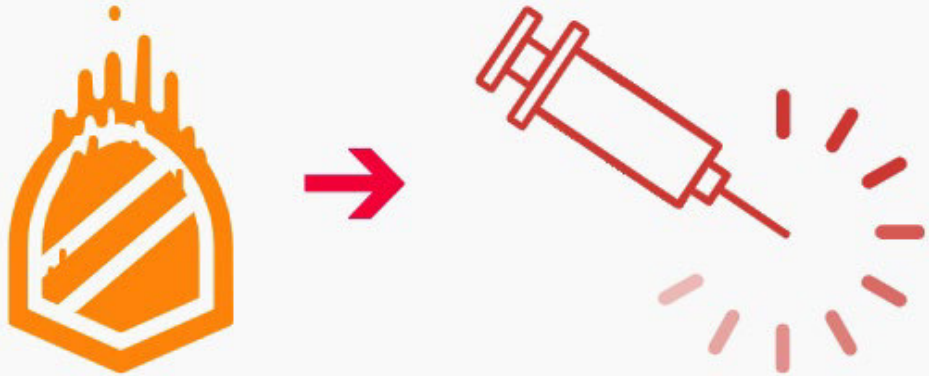
- Meltdown: out-of-reach
- Foreshadow: cache emptied

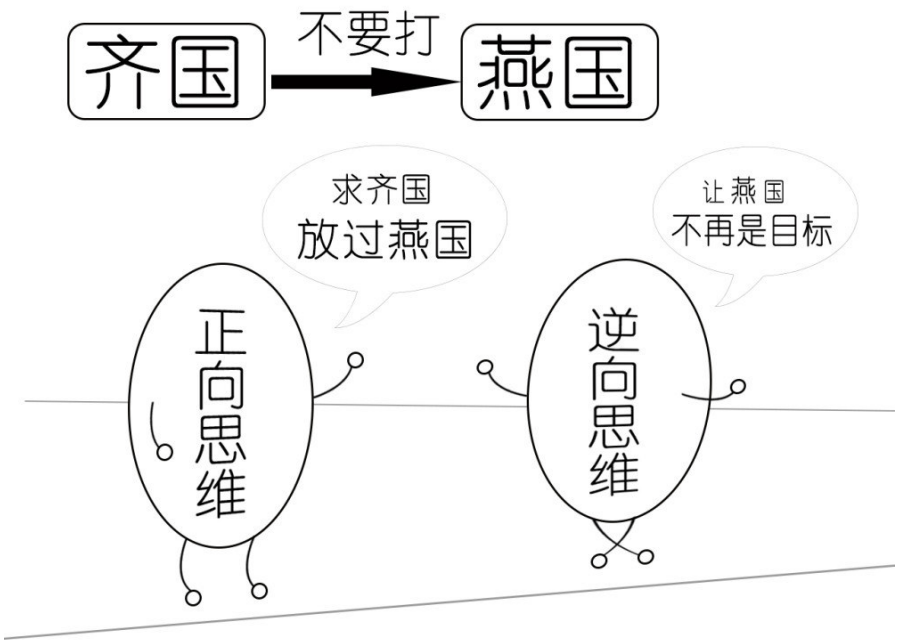
Intra-enclave view

- Access enclave + outside memory

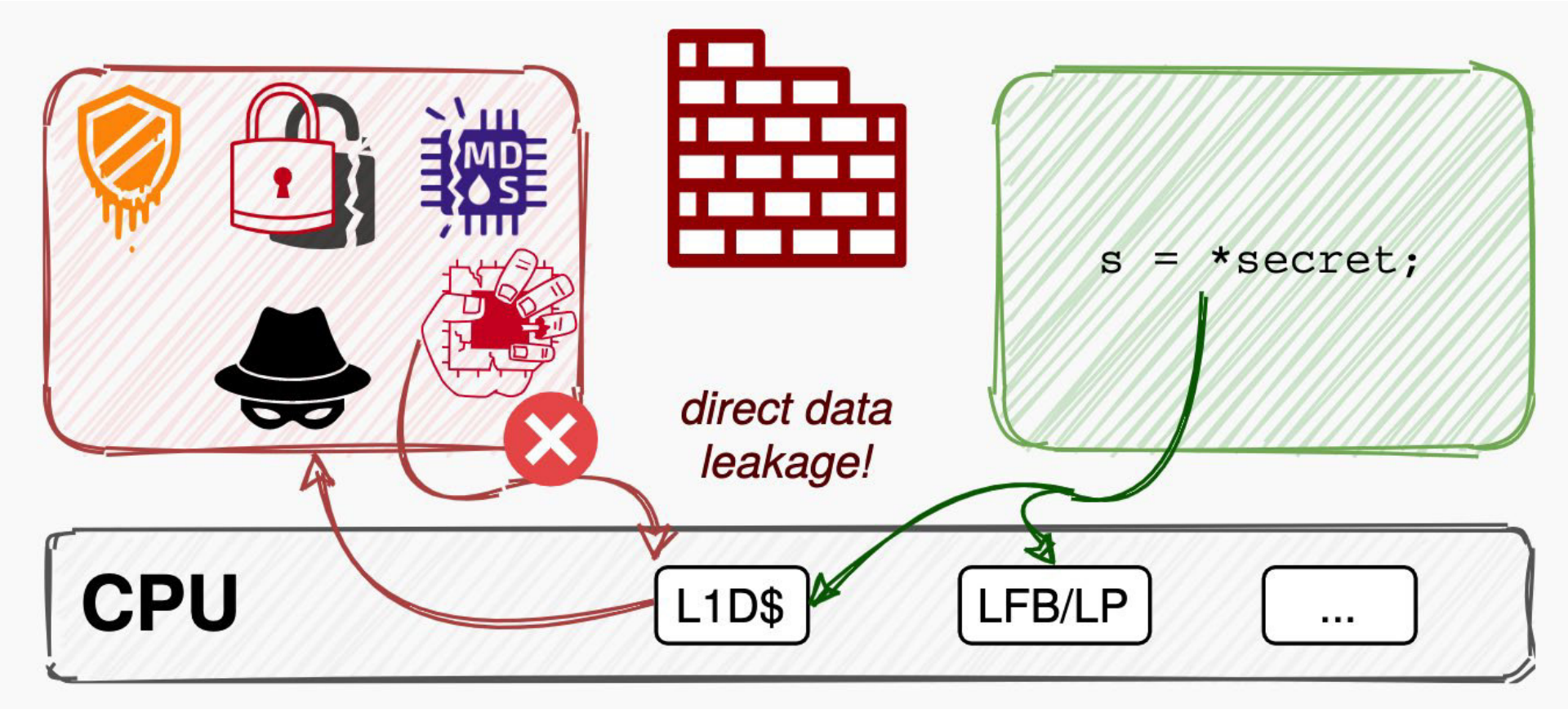
→ Abuse **in-enclave code gadgets!**

Inception：意识注入 逆向思维 (Cont.)

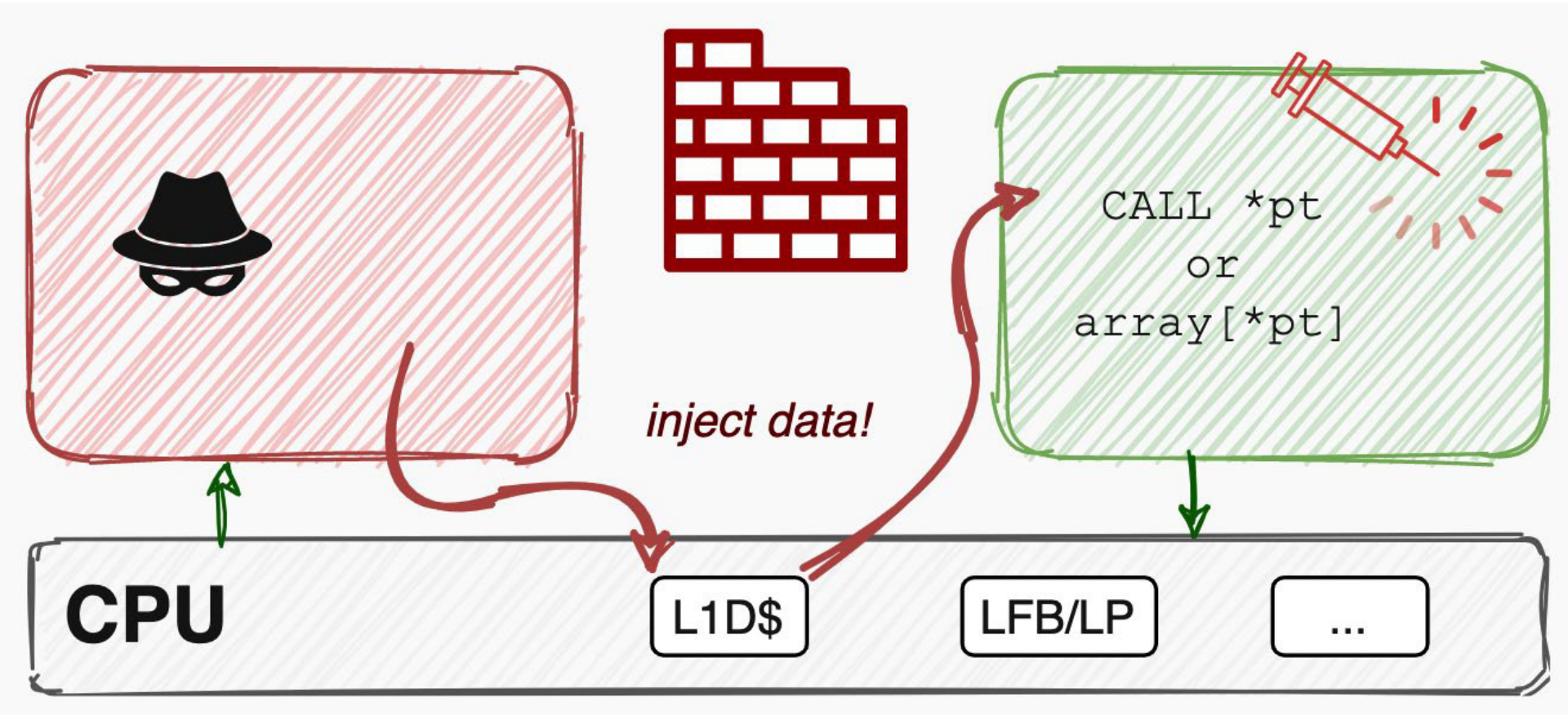




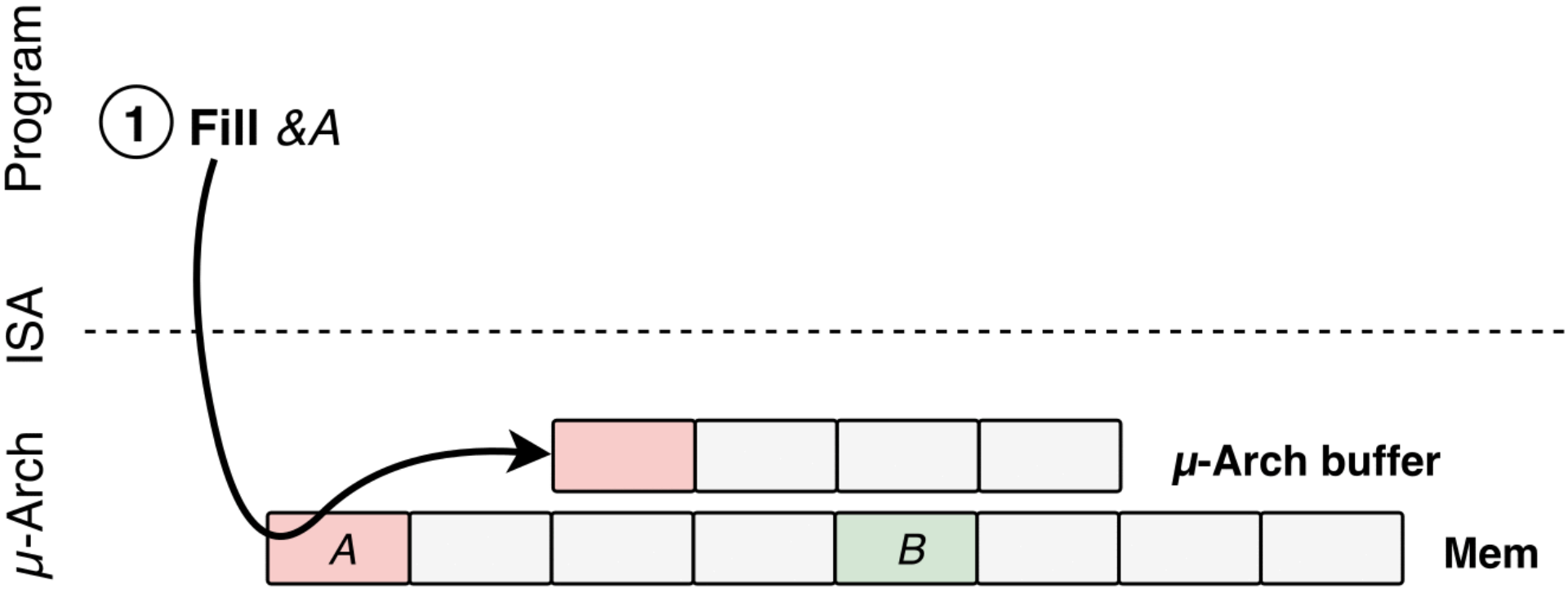
Load Value Injection (LVI): Turning Meltdown around, leaking -> injection



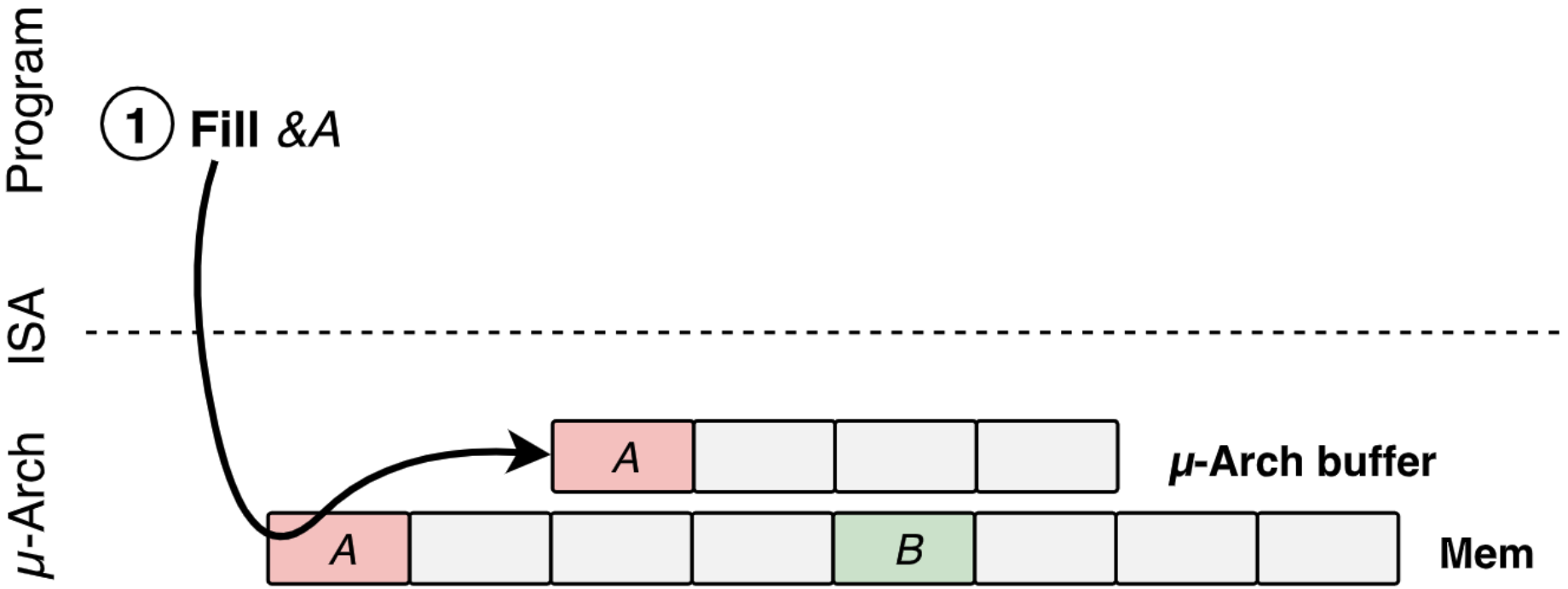
Load Value Injection (LVI): Turning Meltdown around (Cont.)



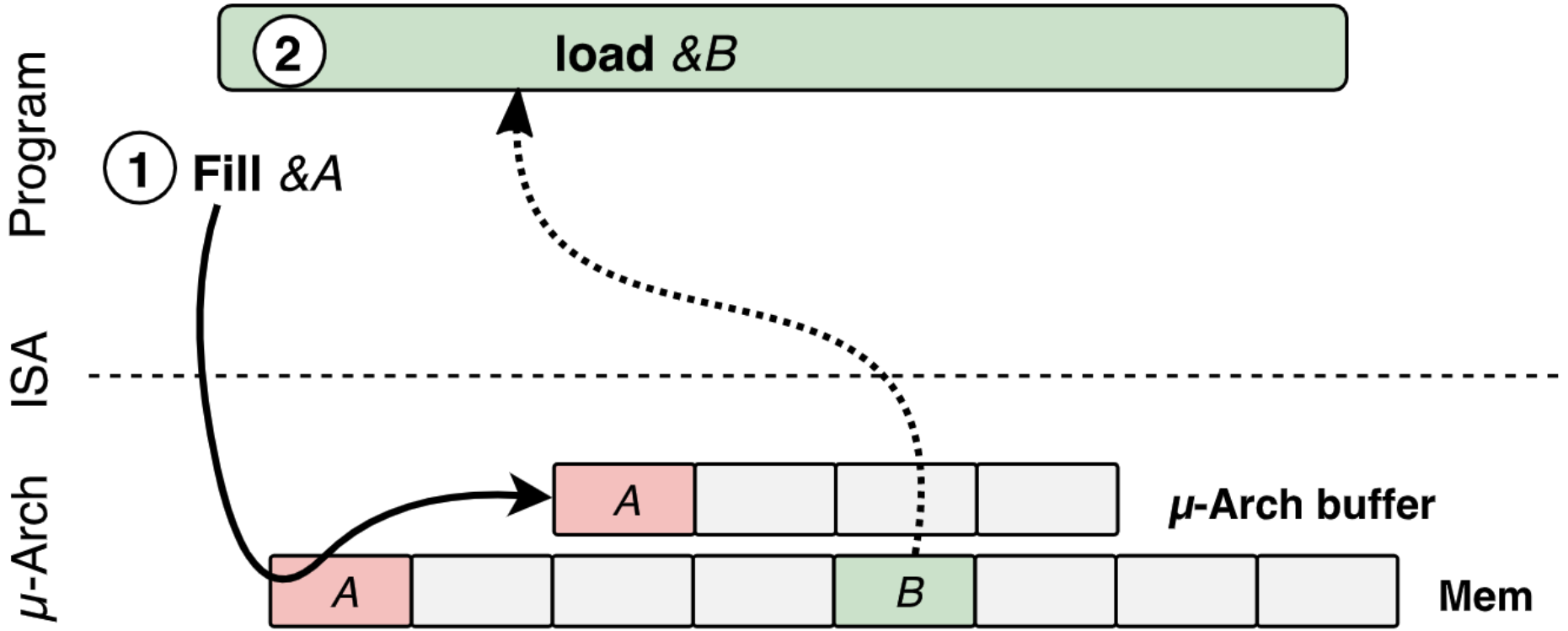
LVI: The basic idea



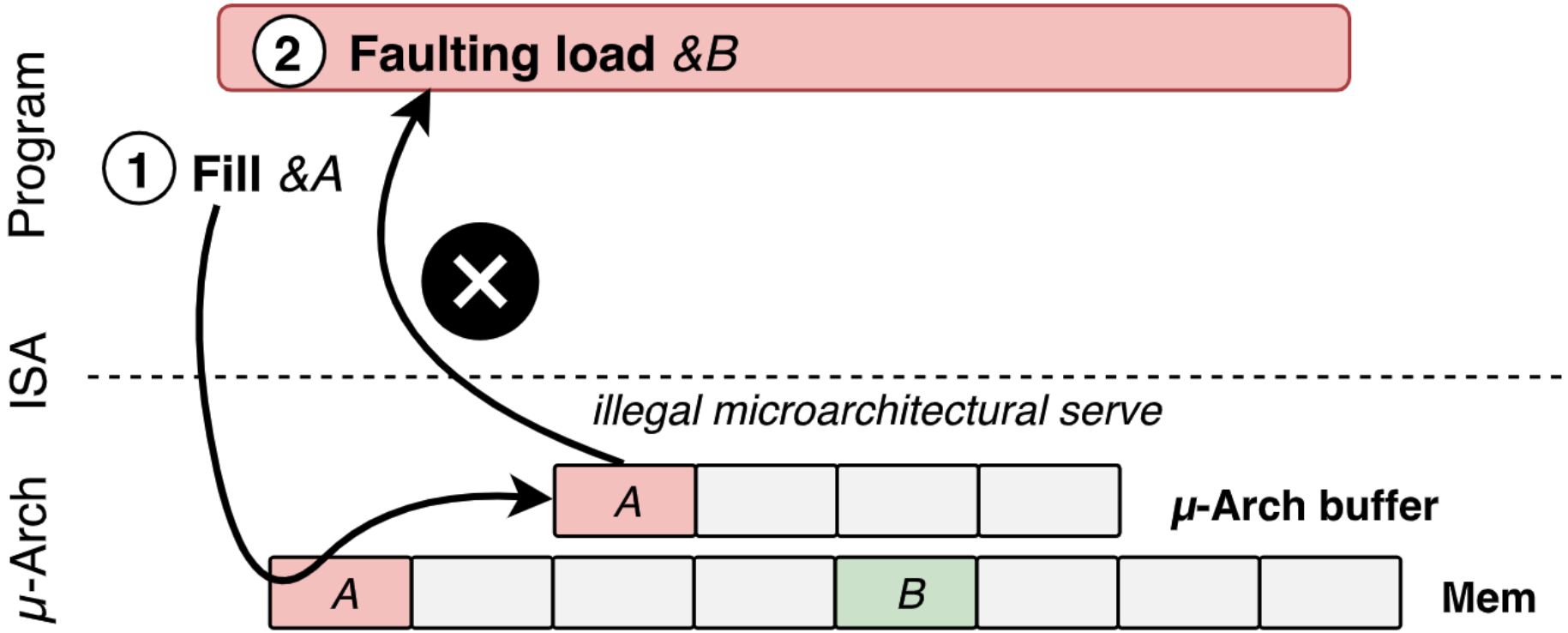
LVI: The basic idea (Cont.)



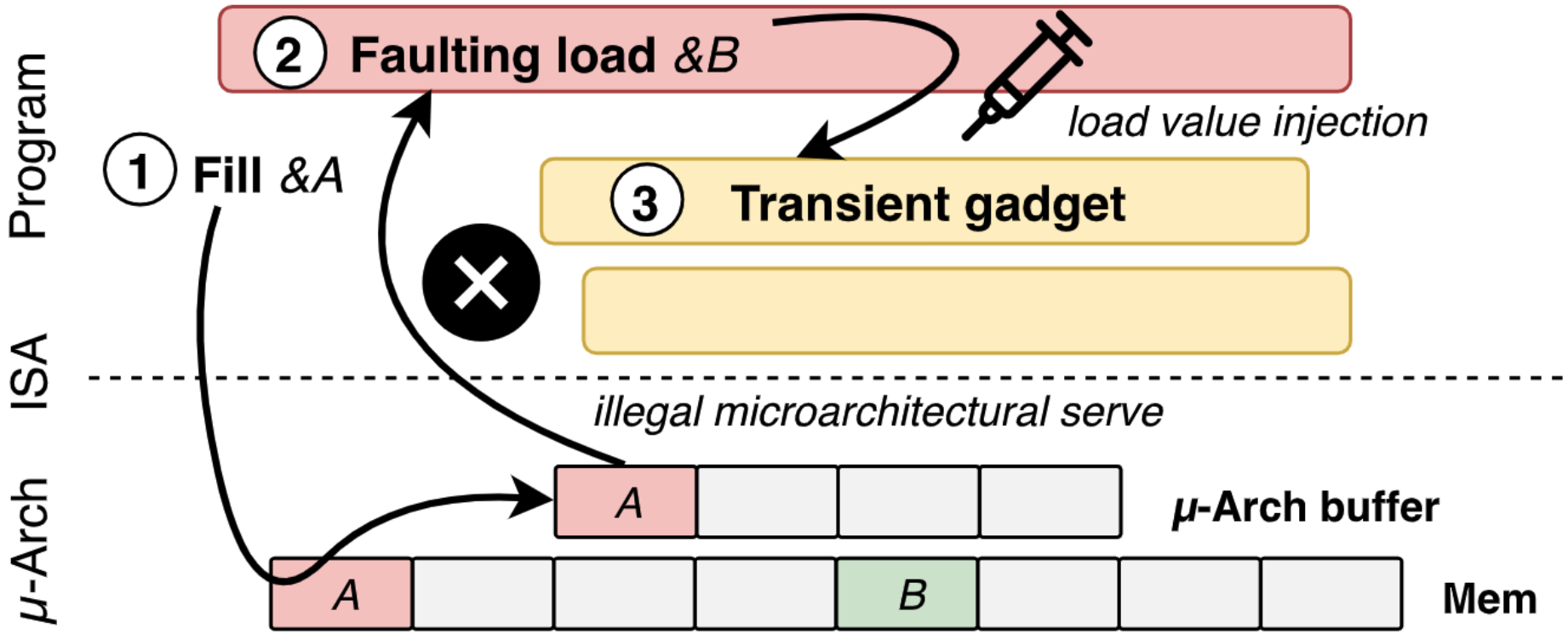
LVI: The basic idea (Cont.)



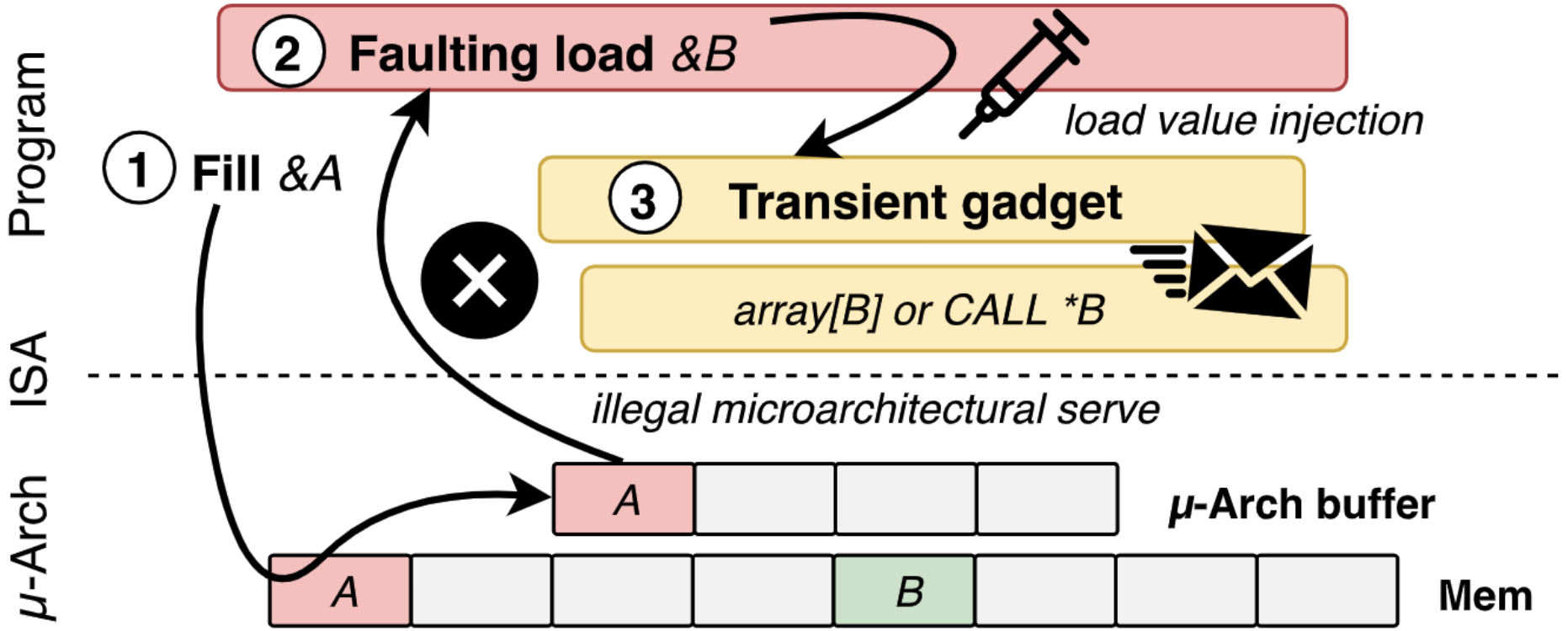
LVI: The basic idea (Cont.)



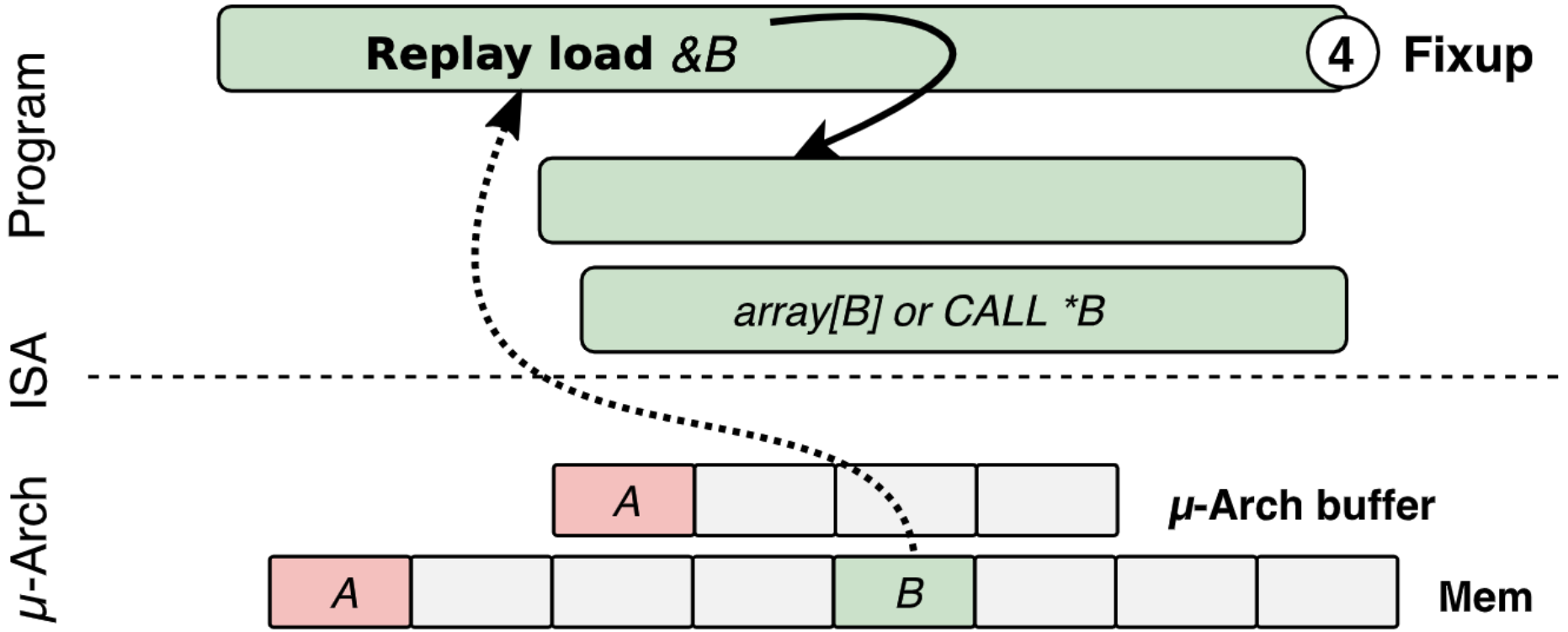
LVI: The basic idea (Cont.)



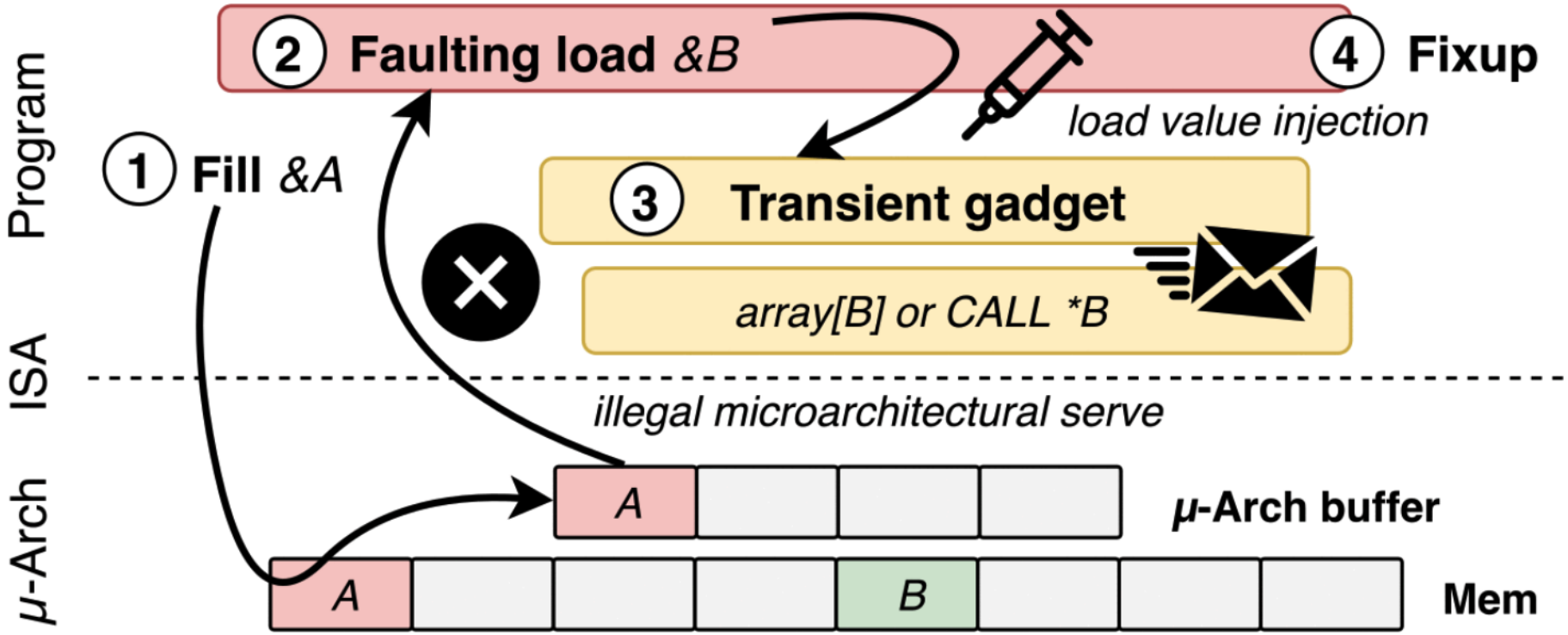
LVI: The basic idea (Cont.)



LVI: The basic idea (Cont.)



LVI: The basic idea (Cont.)



1. 用攻击者的值中毒 (Poison) 隐藏的处理器缓冲区。
2. 在受害程序中引起故障或辅助负载 (Faulting or Assisted load)。
3. 在受害程序加载错误后，攻击者的值会瞬时注入到代码小工具中。
4. 在处理器检测到错误并回滚所有操作之前，侧信道可能会留下与秘密相关的痕迹。

Mechanisms & Implementation

BUILDING BLOCKS OF THE ATTACK

- Phase P1: Microarchitectural Poisoning

  假设逻辑CPU共享竞争缓冲区，精心准备/等待一个CPU微结构状态，此时能够使瞬态forward可控发生，**transient window not long enough**

- Phase P2: Provoking Faulting or Assisted Loads

  引发错误或帮助受害者执行合法和可信的负载

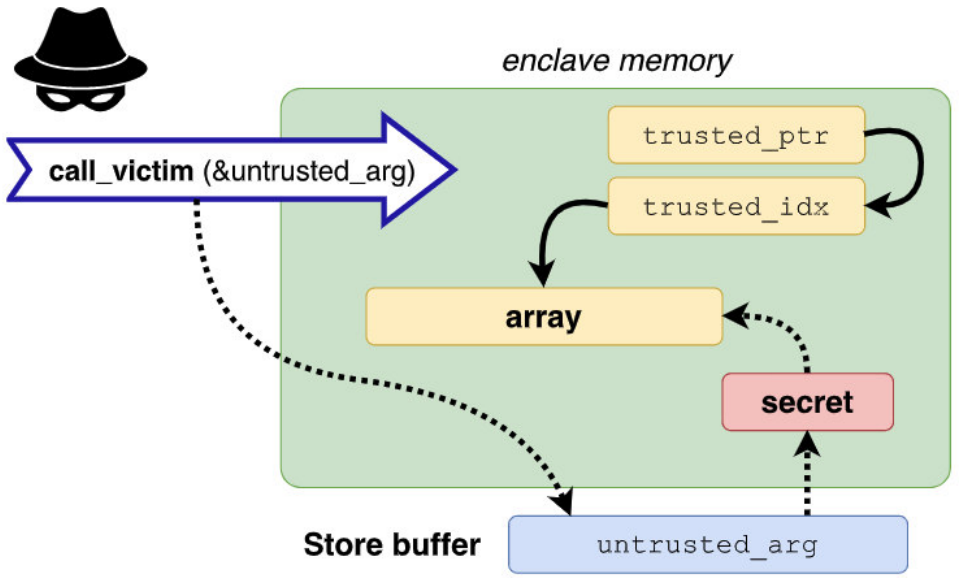
  need a fault where it can also pick up

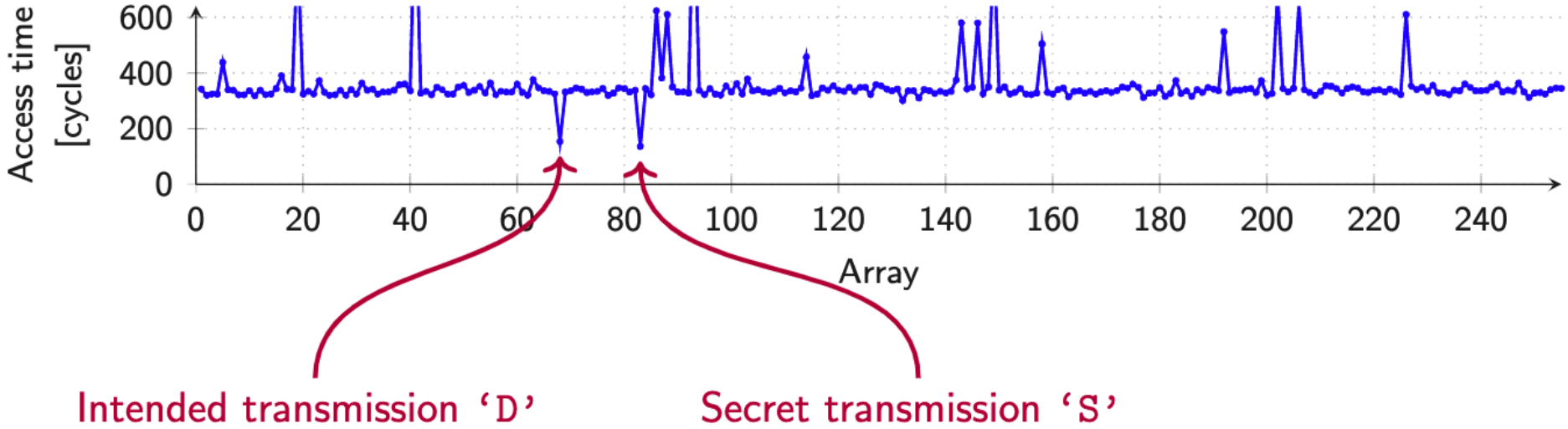
- Phase P3: Gadget-Based Secret Transmission

  要确定一个可重用的代码“小工具”，该小工具对上一阶段从有故障的负载微操作器转发的中毒数据表现出不正确的瞬态行为

LVI toy example: Hijacking transient data flow

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





Classification

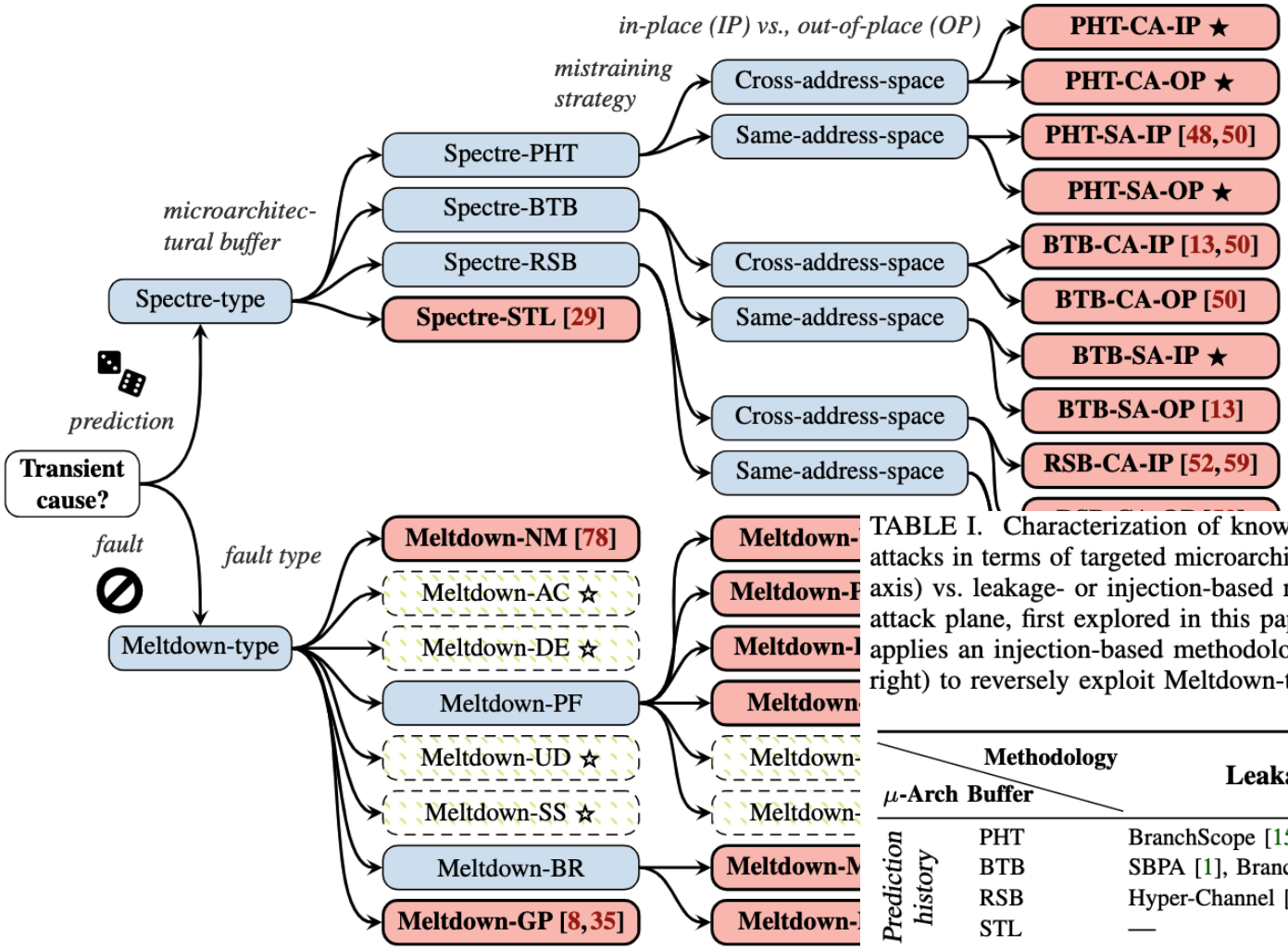


TABLE I. Characterization of known side-channel and transient-execution attacks in terms of targeted microarchitectural predictor or data buffer (vertical axis) vs. leakage- or injection-based methodology (horizontal axis). The LVI attack plane, first explored in this paper, is indicated on the lower right and applies an injection-based methodology known from Spectre attacks (upper right) to reversely exploit Meltdown-type data leakage (lower left).

μ-Arch Buffer \ Methodology		Leakage	Injection
Prediction history	PHT	BranchScope [15], Bluethunder [24]	Spectre-PHT [38]
	BTB	SBPA [1], BranchShadow [40]	Spectre-BTB [38]
	RSB	Hyper-Channel [8]	Spectre-RSB [39, 44]
	STL	—	Spectre-STL [23]
Program data	L1D	Meltdown [42]	LVI-NULL
	L1D	Foreshadow [61]	LVI-L1D
	FPU	LazyFP [57]	LVI-FPU
	SB	Fallout [9]	LVI-SB
	LFB/LP	ZombieLoad [53], RIDL [67]	LVI-LFB/LP

LVI-L1D: L1 Data Cache Injection, control-flow hijacking

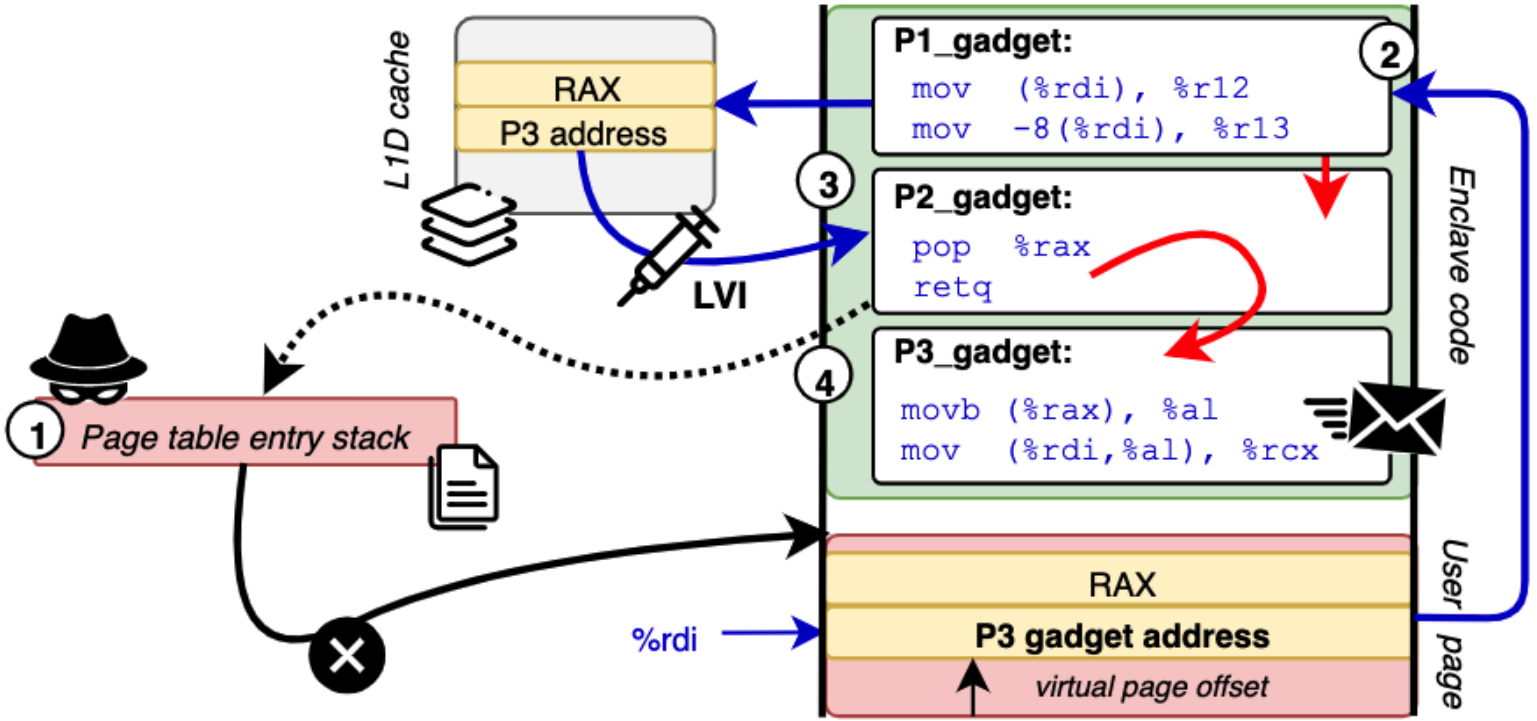
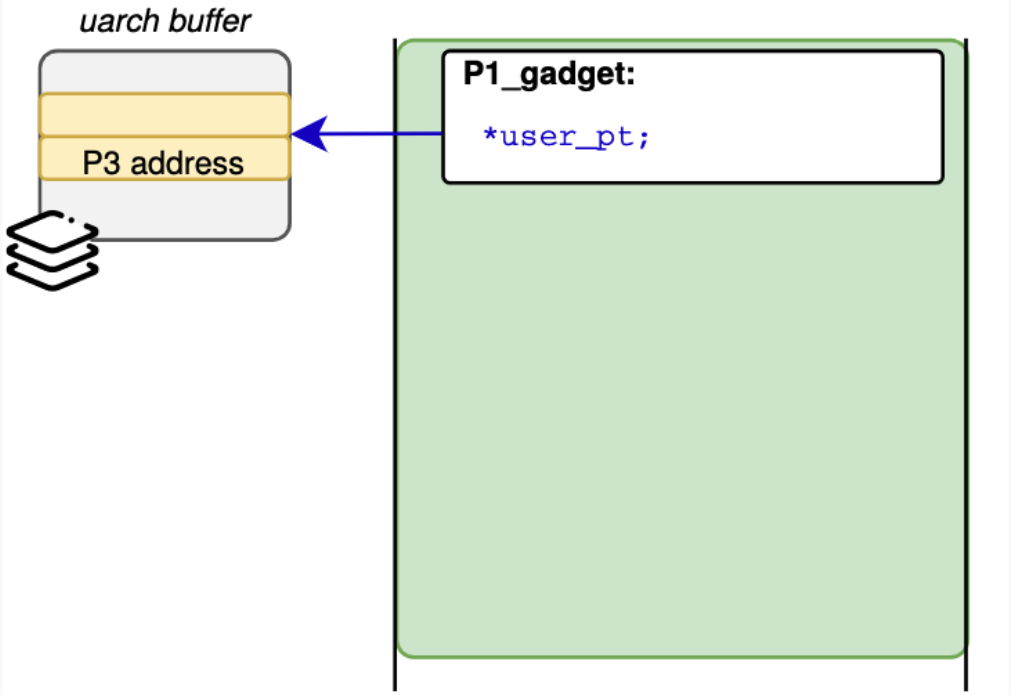


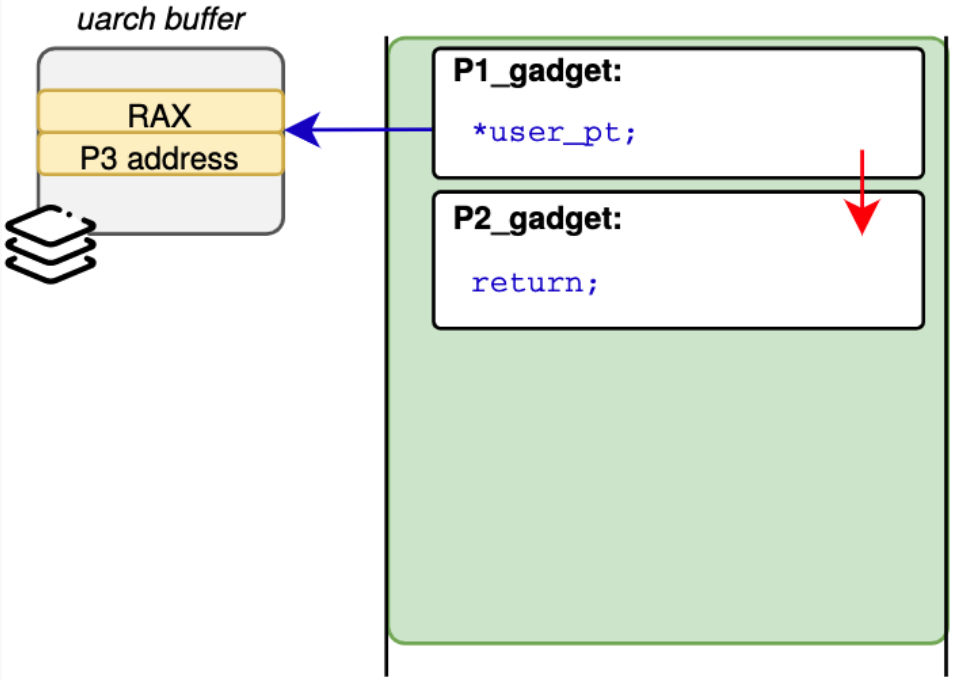
Fig. 5. Transient control-flow hijacking using LVI-L1D: (1) the enclave's stack PTE is remapped to a user page outside the enclave; (2) a $\mathcal{P}1$ gadget inside the enclave loads attacker-controlled data into L1D; (3) a $\mathcal{P}2$ gadget pops trusted data (return address) from the enclave stack, leading to faulting loads which are transiently served with poisoned data from L1D; (4) the enclave's transient execution continues at an attacker-chosen $\mathcal{P}3$ gadget encoding arbitrary secrets in the microarchitectural CPU state.

LVI-based transient control-flow hijacking



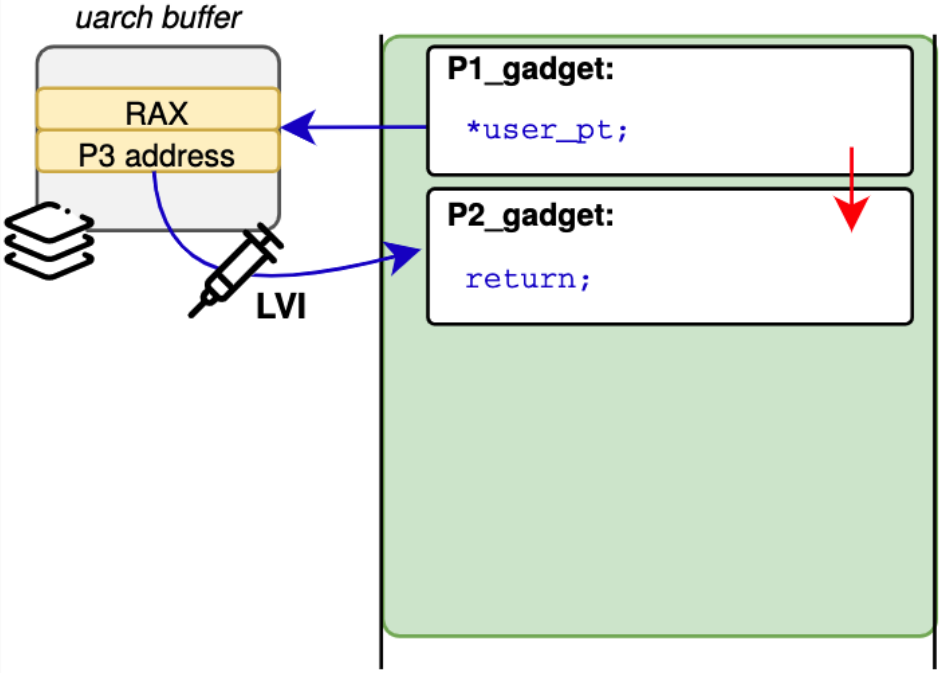
1. Victim fills μ-arch buffer with attacker-controlled data

LVI-based transient control-flow hijacking (Cont.)



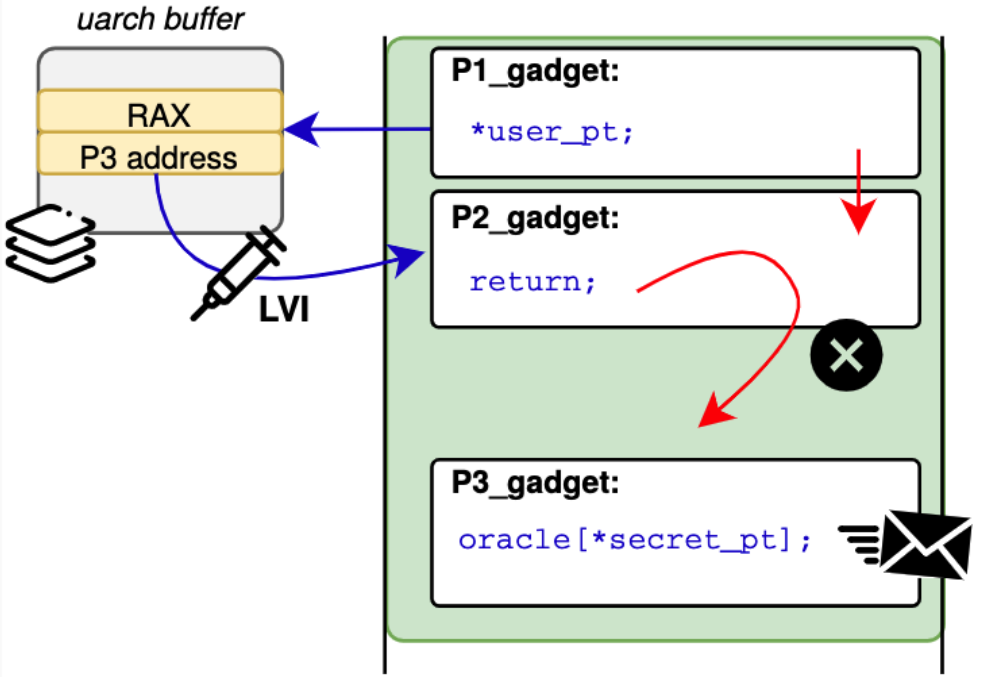
1. Victim fills μ-arch buffer with attacker-controlled data
2. Victim executes indirect branch (JMP/CALL/RET)

LVI-based transient control-flow hijacking (Cont.)



1. Victim fills μ-arch buffer with attacker-controlled data
2. Victim executes indirect branch (JMP/CALL/RET)
3. Faulting load $\rightarrow$ inject incorrect attacker values(!)

LVI-based transient control-flow hijacking (Cont.)



1. Victim fills μ-arch buffer with attacker-controlled data
2. Victim executes indirect branch (JMP/CALL/RET)
3. Faulting load $\rightarrow$ inject incorrect attacker values(!)
4. Redirect transient control flow

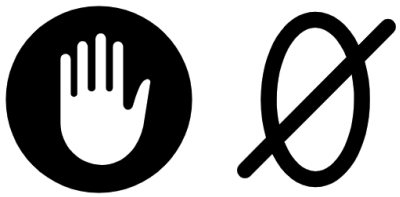
LVI-SB, LVI-LFB, and LVI-LP: Buffer and Port Injection (SGX & non-SGX)

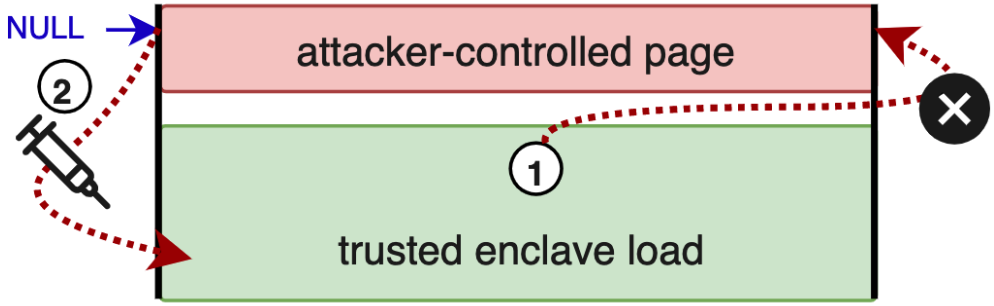
```
1 ; %rbx: user-controlled argument ptr (outside enclave)
2 sgx_my_sum_bridge:
3     ...
4     call my_sum              ; compute 0x10(%rbx) + 0x8(%rbx)
5     mov %rax,(%rbx)          ; P1: store sum to user address
6     xor %eax,%eax
7     pop %rbx
8     ret                      ; P2: load from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

We can setup a fake transient stack in the store buffer or L1D!

- 攻击者可以在#4后中断安全区，OS清除堆栈或已访问的位，然后恢复安全区；
- 当安全区代码在第8行返回时，控制流将重定向到攻击者注入的位置，因为故障或Assistant ret 错误地从Store Buffer (SB) 提取了该注入值
- 攻击者可以通过在受害者安全区代码中将一个或多个P3小工具链接在一起来对任意安全区秘密进行编码

LVI-NULL: Why 0x00 is not a safe value, function-pointer hijacking





- 0x00 dummy values总被作为错误后的ret值
- 但是NULL是一个有效的虚拟内存地址，且在攻击者的控制下
- 可劫持该指针形成注入可能

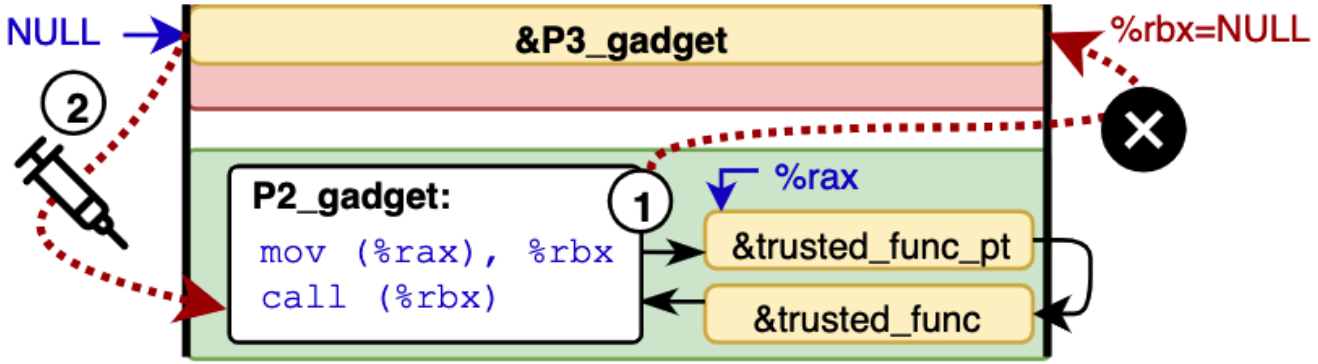


Fig. 6. Transient control-flow hijacking using LVI-NULL: (1) a $\mathcal{P}2$ gadget inside the enclave dereferences a function pointer-to-pointer, leading to a faulting load which forwards the dummy value null; (2) the following indirect call transiently dereferences the attacker-controlled null page outside the enclave, causing execution to continue at an attacker-chosen $\mathcal{P}3$ gadget address.

Novelty

创新要素

- 逆向思维
  - 站在巨人的肩膀上（前面的攻击方案）
- 看《盗梦空间》，用科学家的眼光，会注重其搭载的想象力
- 植入思想，高维打击
- 旧瓶装新酒，触类旁通
- 网络战四大手段：断、截、伪、改

Source: http://blog.sciencenet.cn/blog-287179-379573.html

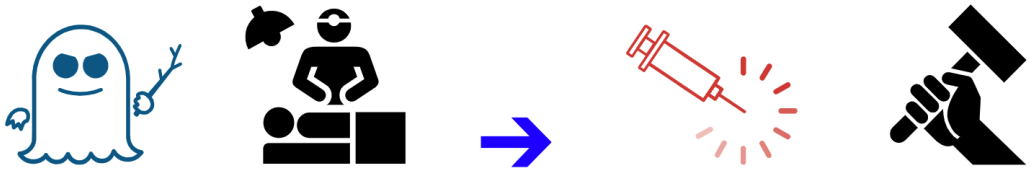
Mitigation & Evaluation

影响范围与程度：主要针对Intel

- 主要影响Intel SGX的特定工作，损害其生态系统中的远程认证保障(QE)
- 但其他易受Meltdown攻击的处理器也可能也易受LVI（AMD和ARM）的攻击。
- 有趣的是，IBM POWERPC不受影响。

Vulnerable platforms: Intel Software Guard Extensions (SGX)

intel · Microsoft Azure · Signal · Enarx (Red Hat) · Asylo (Google) · wolfSSL · Fortanix · DELL



23 fences

October 2019—"surgical precision"

49,315 fences

March 2020—"big hammer"

Intel architectural enclaves: lfence counts
[libsgxqe.signed.so]

同步的独立CVE-2020-0551

- LVI最早由Jo Van Bulck于2019年4月4日发现并通告，Intel认为问题不大，下述PoC提交后，才fix，后发表相应学术论文；
- Bitdefender的研究人员独立发现了LVI-LFB (CVE-2020-0551)，并在2020年2月向Intel提供了一个显示LVI-LFB在跨进程和超线程场景中的PoC

Source: https://software.intel.com/security-software-guidance/processors-affected-transient-execution-attack-mitigation-product-cpu-model

LVI-ROP in SGX Quoting Enclave攻击效果

```
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
```

Listing 4: LVI gadget in SGX-SDK `intel_fast_memcpy` used in QE.

a benchmark enclave on an i7-8650U with the latest microcode 0xb4

- 最理想：在victim enclave中禁用堆栈随机化，LVI攻击非常成功，从10万次运行中获取了99453次注入值，平均每秒9090次尝试，公开小工具实现了无错误的传输速率，为**9.04 kB / s**。
- 常规环境：包括堆栈随机化内的防御手段均开启，攻击成功效果平均下降128倍。在10万次运行中，注入的返回地址被读写776次，导致传输速率为**70.54 B / s**。而且对官方签名的QE进行攻击，仅能打开缺口，实际用途不是很大，但是 Intel 还是进一步用软件方法加固了TCB，侧面说明有效性。

LVI-NULL fault injection to reduce AES rounds

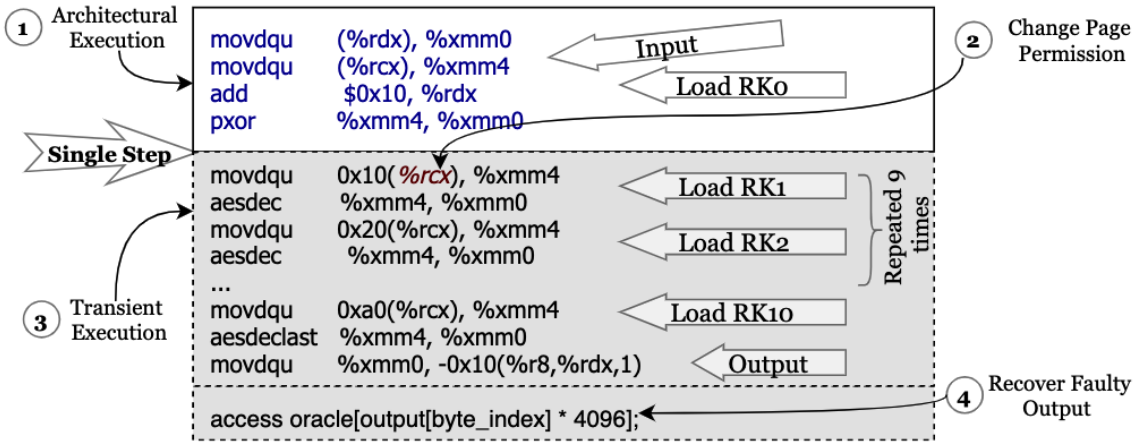


Fig. 7. Overview of the AES-NI fault attack: (1) the victim architecturally executes the initial AES round, which `xors` the input with round key 0; (2) access rights on the memory page holding the key schedule are revoked; (3) upon the next key access ($\mathcal{P}2$), the enclave suffers a page fault, causing the CPU to transiently execute the next 10 AES rounds with zeroed round keys; (4) finally the faulty output is encoded ($\mathcal{P}3$) through a cache-based side-channel.

- 使用RDCL_NO和microcode 0xae在 i9-9900K 上针对100个不同的AES密钥进行了攻击；
- 平均而言，每个恢复的Key候选有83%的时间与预期的错误输出匹配；
- 从多数的10个Key，AES块的16个字节中恢复了平均15.61个正确输出，匹配度为**97%**；
- 平均攻击时间为**25.94 s**（包括安全区创建时间），并且需要**246707**次AES函数的执行，成功地恢复了零位秘密密钥。

LVI的其他攻击面

- 对传统user-kernel隔离机制的有效性
  - 实际上页面替换时利用该思想可从用户态高效猜解读出kernel
  - LVI-SB i7-8650U with Linux kernel 5.0 下，每7739个（n = 100 000）辅助内核中的1个使用存储缓冲区中注入的值代替体系结构上正确的值。平均每隔**6.5 s**就成功地将一个值注入内核执行中。如果关闭上下文切换时检查，每8个（n = 100 000）辅助内核中就有1个。
- 跨进程注入攻击
  - Windows会定期清除PTE访问的位，攻击者再次使用 clflush 从缓存中清除目标共享内存位置；
  - Intel i7-8650U with Linux kernel 5.0 下，101个辅助负载（n = 100 000）中，有1个使用了攻击者注入的值，导致注入概率接近1%。通过平均每秒1122次尝试，LVI 公开小工具的传输速率达到了**11.11 B / s**。

缓解思路

- 硬件：长期、有效、完全解决方案、成本高
- 软件：短期、临时解决方案、trade off
  - **与以前的Meltdown类型攻击相反，处理器微代码更新以刷新受影响的缓冲区已不再足够**

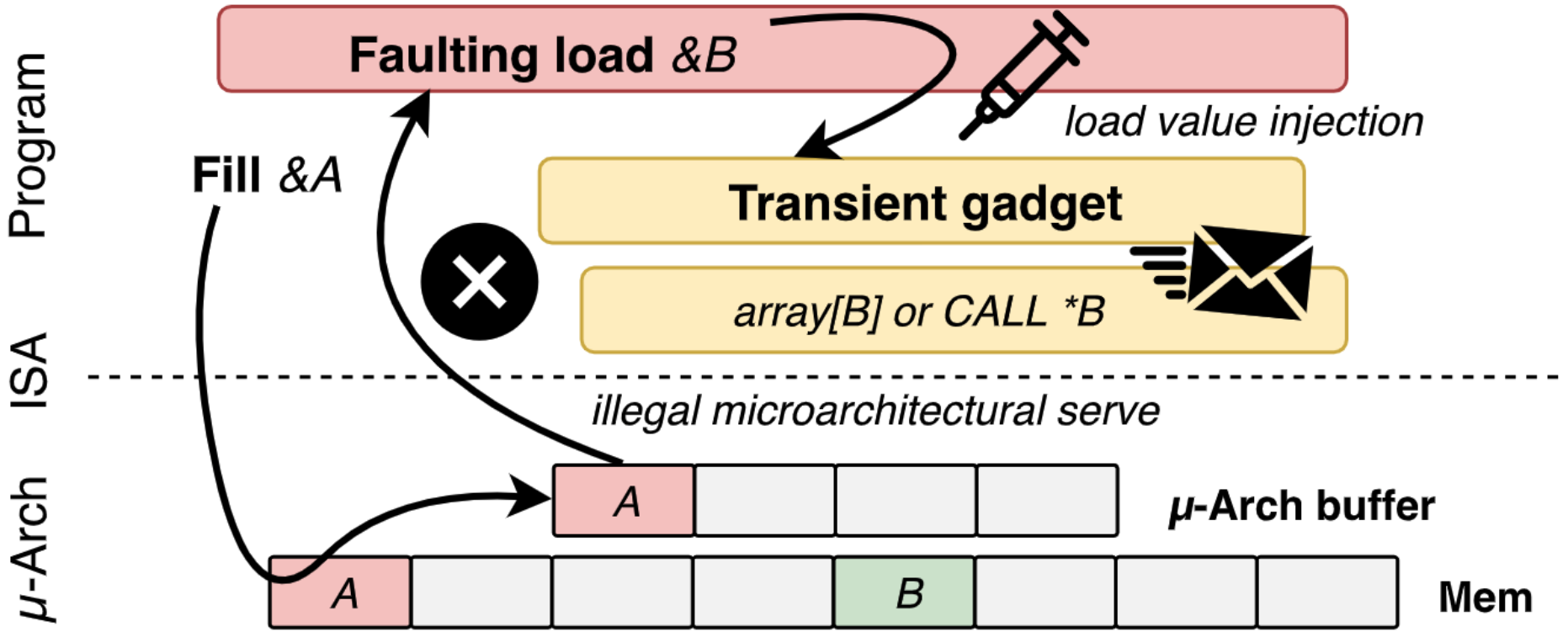
TABLE II. Indirect branch instruction emulations needed to prevent LVI and whether or not they require a scratch register which can be clobbered.

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`	✓

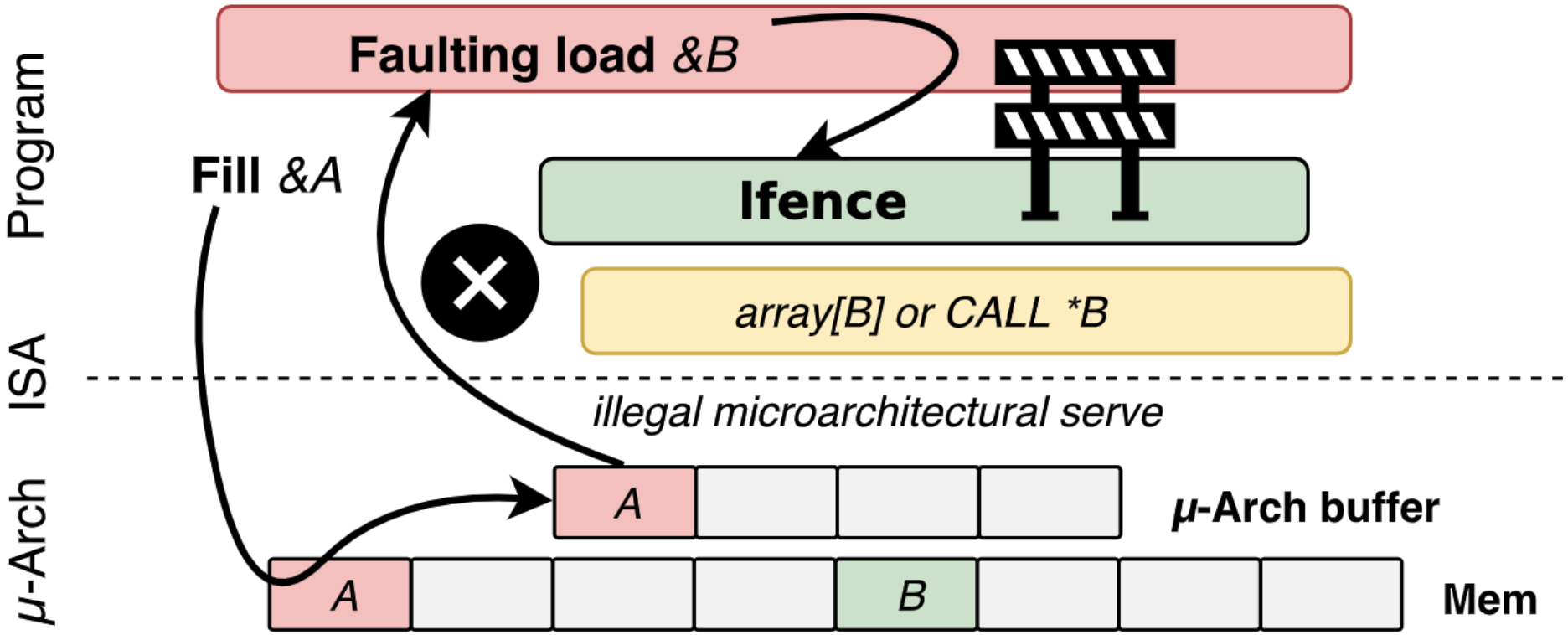
缓解思路：软件方案

Mitigating LVI?

Gadget-based exploits → flushing μ-arch buffers does not suffice!



缓解思路：软件方案 (Cont.)



- LVI要求编译器补丁插入显式的 lfence
- 该壁垒在潜在的每条易受攻击的加载指令之后对处理器管道进行序列化
- 由于隐式load，某些指令必须列入黑名单，其中包括无处不在的x86 ret指令

行动上仍纷纷给出patch

-mlfence-after-load

GNU Assembler Adds New Options For Mitigating Load Value Injection Attack

Written by Michael Larabel in GNU on 11 March 2020 at 02:55 PM EDT. 14 Comments

-mlvi-hardening

LLVM Lands Performance-Hitting Mitigation For Intel LVI Vulnerability

Written by Michael Larabel in Software on 3 April 2020. **Page 1 of 3**. 20 Comments

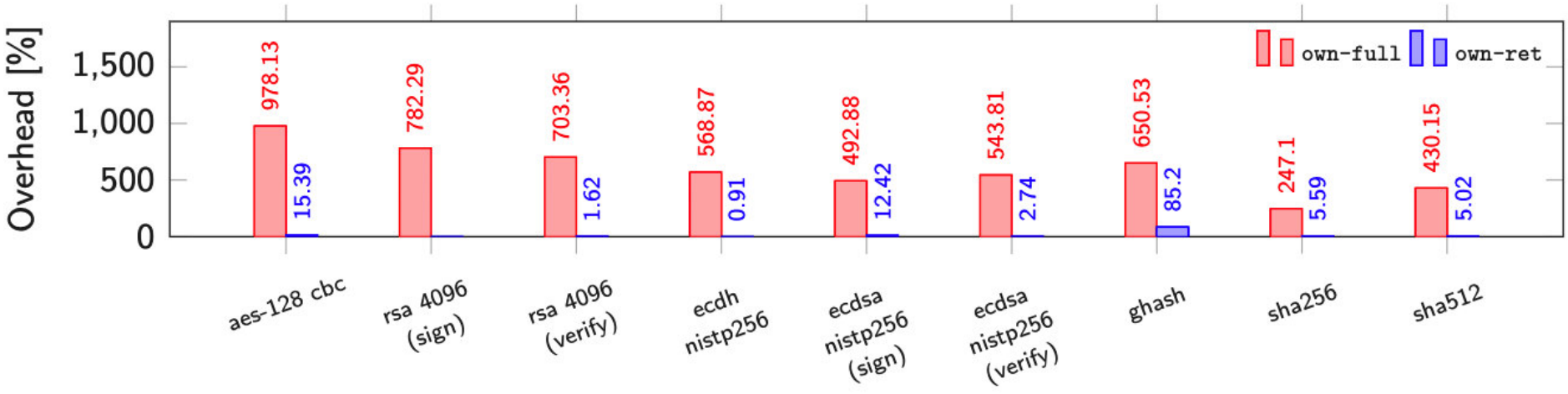
Visual C++

-Qspectre-load

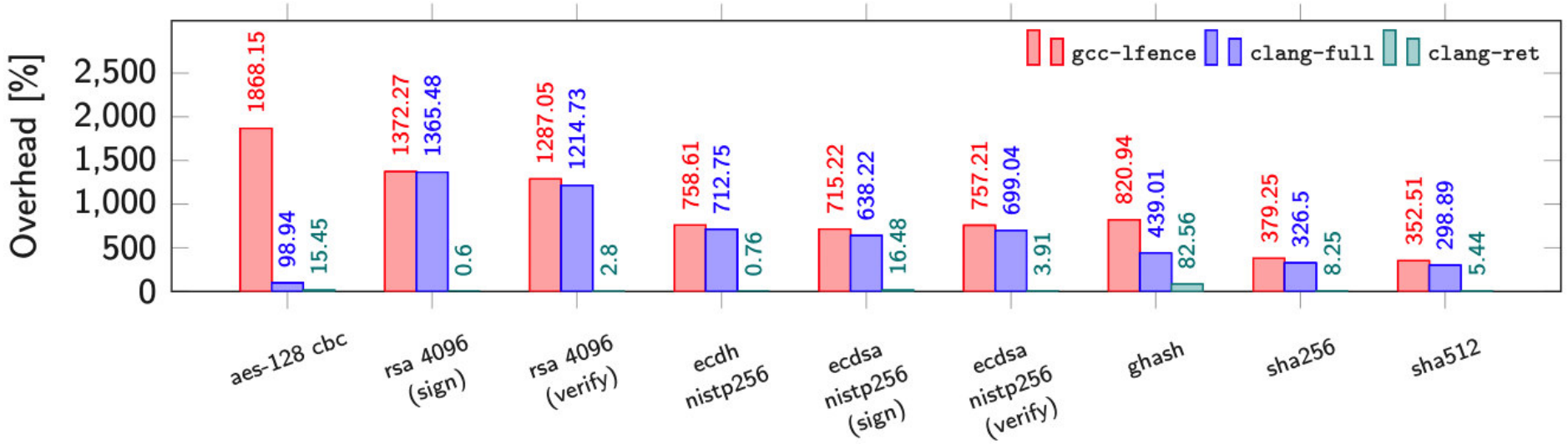
More Spectre Mitigations in MSVC

March 13th, 2020

Performance overheads: OpenSSL



(author's prototype mitigation)



(Intel's mitigation)

Performance overheads: SPEC 2017 (Intel's mitigation)

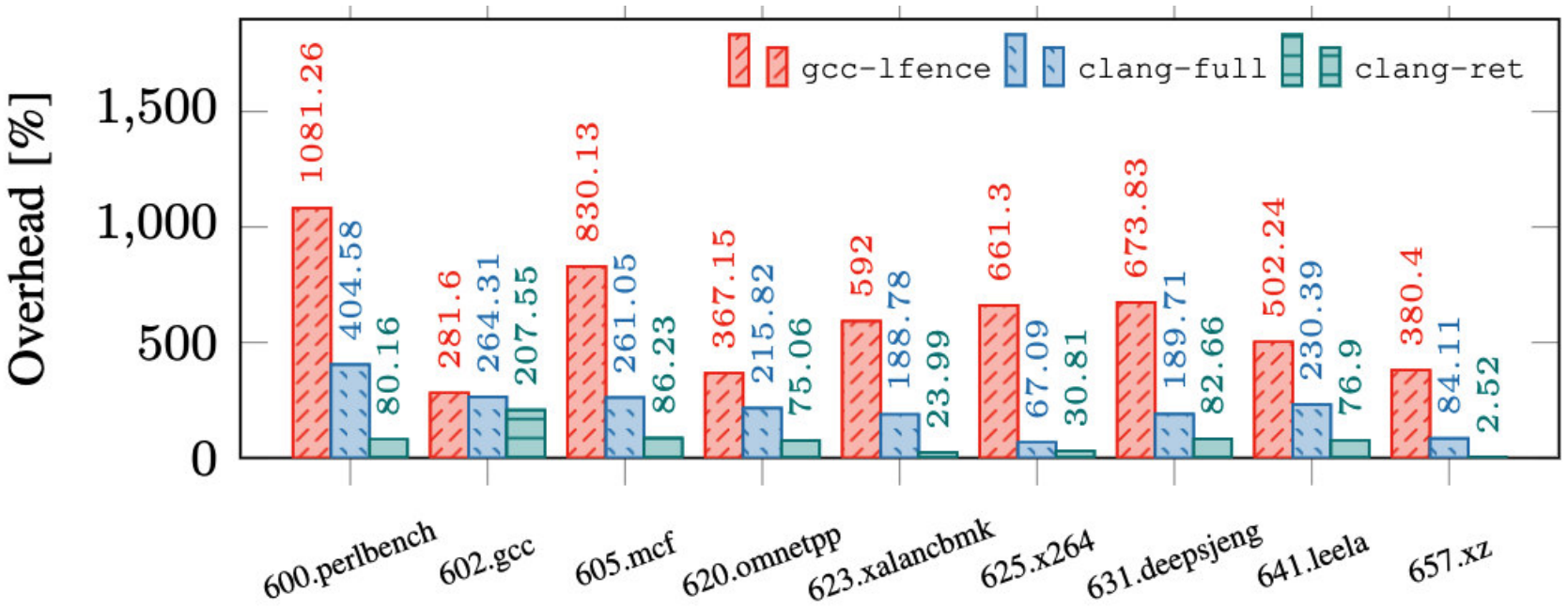


Fig. 9. Performance overhead of Intel's mitigations for non-optimized assembler `gcc` (fence loads + ret) and optimized `clang` (fence loads + indirect branch + ret vs. ret-only) for SPEC2017 on an Intel i9-9900K CPU.

Strengths & Weaknesses

Strength

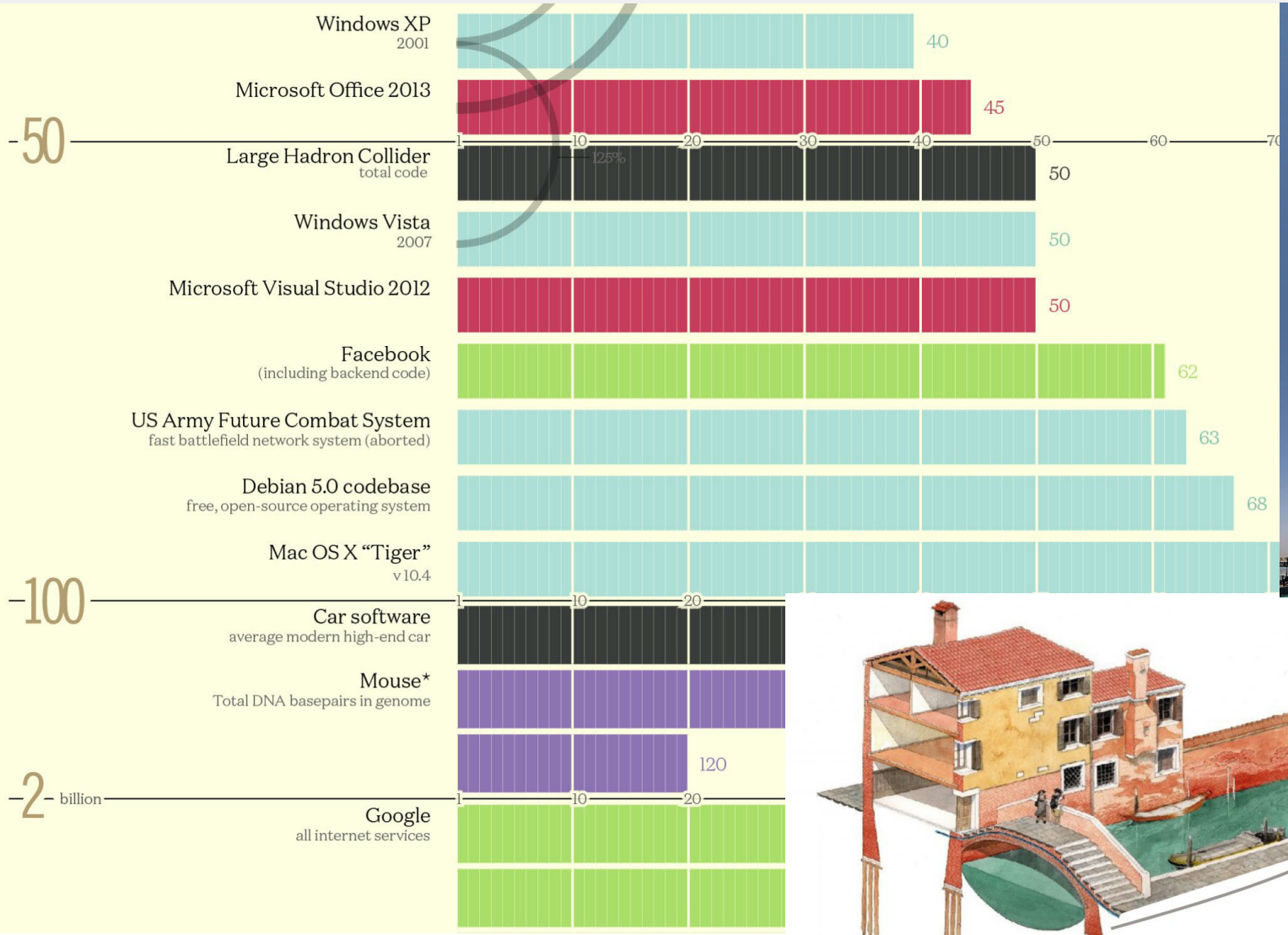
- 新兴而强大的瞬态执行攻击类，开新坑，我们有更多研究机遇
- 对基础的侧信道研究很重要
- 安全性上打穿了系统栈：硬件、管理程序、内核、编译器、应用程序
- a very subtle art
- 工作十分充分
- 论文结构很清晰，多次点题
- Simple, novel mechanism to solve an important problem
- 有HW/SW Co-Design
- 强有力的攻击，攻击事前事后均难以被察觉

Weakness

- 目前侮辱性极强，但可利用率不高；
- 主要针对intel
- 真正实现攻击时需要天时地利人和
- 攻击前成本较高，且一旦被针对即失效
- transient window not long enough
- need a fault where it can also pick up the value

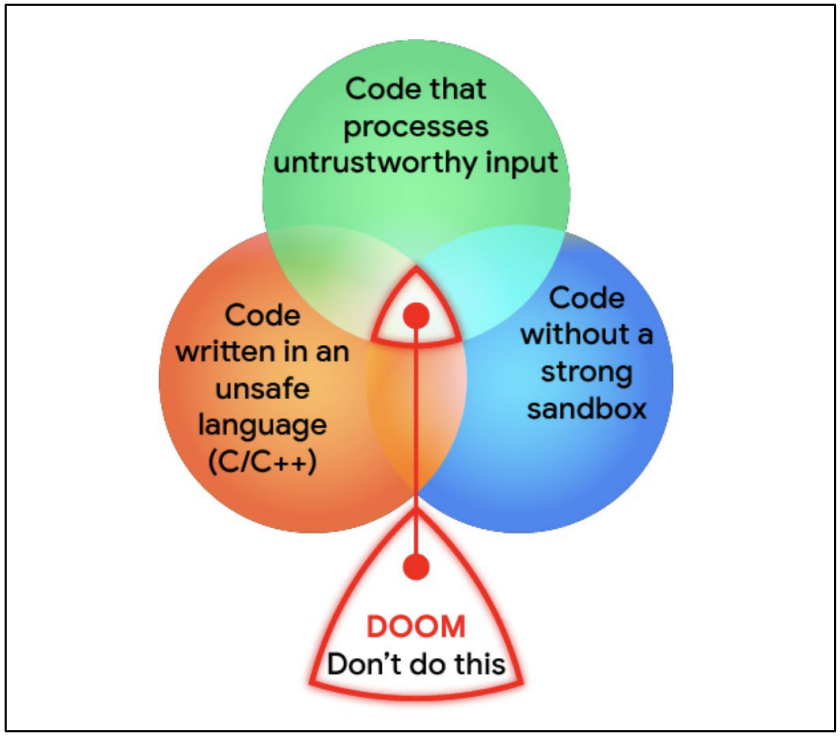
Discussion

From architecture. . . to microarchitecture



Source: https://en.wikipedia.org/wiki/Burj_Khalifa

还有哪些空白可探索



Source: https://www.slideshare.net/paulgoogle/20150326-02

Background：

- **现有CPU攻击，Meltdown等不够心机，有效率不够**
  - ▶ 分支预测有侧信道
  - ▶ 攻击模式可结合，以绕过现有缓解方案

Key Idea：

- **逆向思维，高维打击。配合精心构建攻击方案实现由内而外的核打击**
- **与其将数据直接从受害者身上泄漏到攻击者，不如通过隐藏的处理器缓冲区将不正确的数据注入到受害者，再劫持瞬态执行**

Mechanisms ：

- **四步攻击：注入、读取、执行、hack，具体实现多种多样**

Results ：

- **大规模型号CPU受影响，从Intel SGX窃取敏感数据和密钥**
- **硬件解决，成本极高；软件临时缓解，计算速度降低2-19倍**
- **新型的攻击思路，但实际利用时难度极高**

感谢批评指正

THANKS