black hat EUROPE 2024

DECEMBER 11-12, 2024

BRIEFINGS

Improving Side-Channel Protections for Intel® TDX

*Scott Constable	Defensive Security Researcher, Intel Labs, Intel Corporation
*Nagaraju (Raju) Kodalapura	Principal Engineer, Offensive Security Research, Intel Corporation
Baruch Chaikin	Principal Engineer, CPU Architecture, Intel Corporation

*Speakers

Acknowledgements

- Intel Labs
- Intel Product Security and Assurance (IPAS) research
- Intel CPU Security Effectiveness Team
- Intel CPU Architecture Team
- The TDXdown researchers at TU Lübeck: Luca Wilke, Florian Sieck, Thomas Eisenbarth

Scott Constable

Nagaraju (Raju) Kodalapura

Baruch Chaikin

Agenda

- Intro to Confidential Computing, Intel TDX (Trust Domain Extensions), side-channel Attacks, and malicious single-stepping
- Pre-TDX PoC (Proof of Concept) TDX-step exploit and mitigation
- Techniques to bypass the TDX-Step mitigation, and intro to the new ICSSD (Instruction Counting Single-Step Defense) feature
- Comparison with the SGX-Step mitigation

What is Confidential Computing (CC)?

Protects **data at rest** (in storage, a database, etc.).

- Data encryption
- Access control

Protects **data in transit** (over a network, PCI bus, etc.)

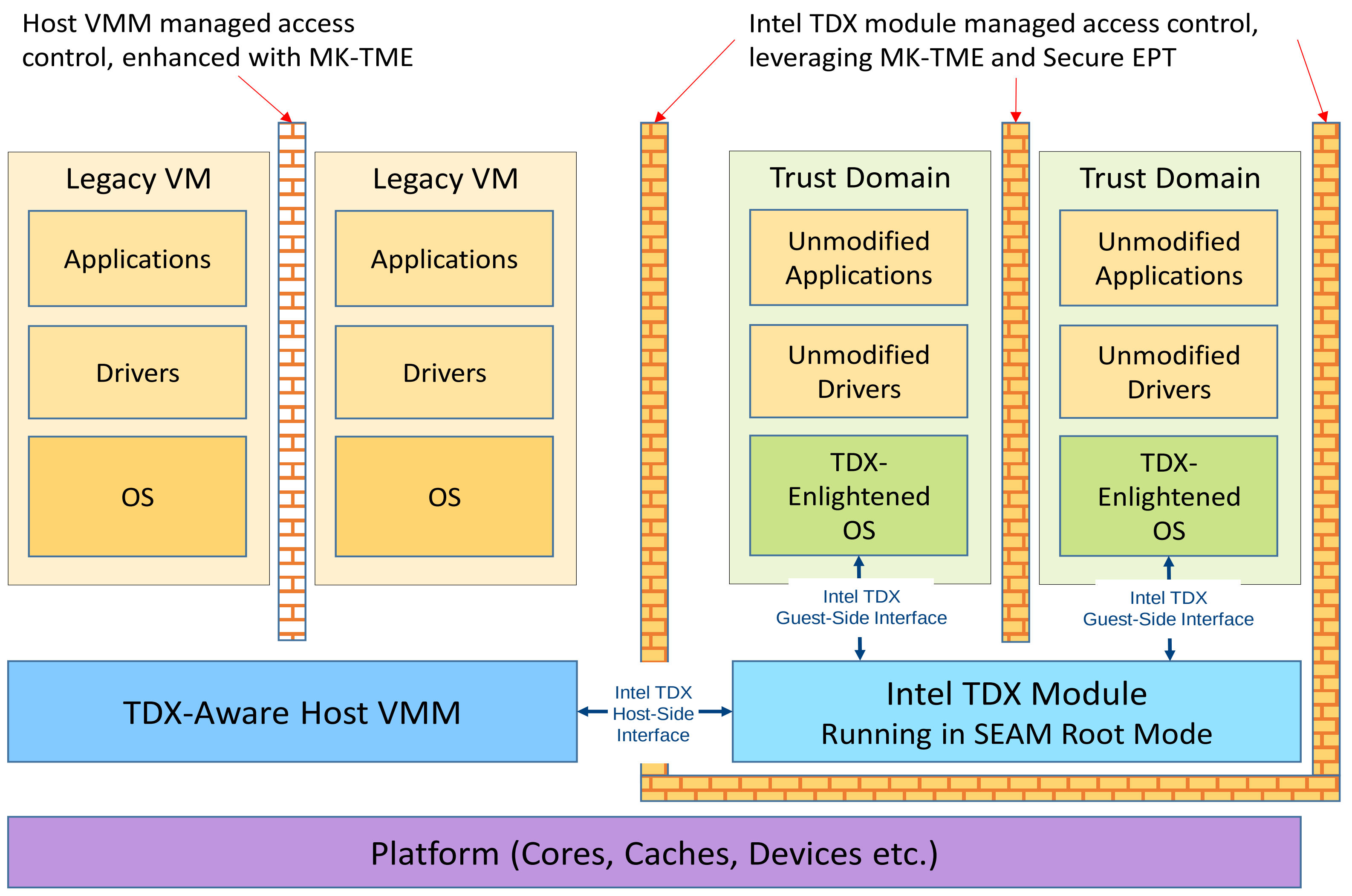
- HTTPS
- TLS

The Focus of CC

Protects **data in use** (within a CPU, XPU, etc.)

- Hardware-based, attested Trusted Execution Environments (TEEs) such as Intel TDX and Intel SGX

What is Intel® TDX?


Host VMM managed access control, enhanced with MK-TME
Intel TDX module managed access control, leveraging MK-TME and Secure EPT
Legacy VM
Applications
Drivers
OS
Legacy VM
Applications
Drivers
OS
Trust Domain
Unmodified Applications
Unmodified Drivers
TDX-Enlightened OS
Intel TDX Guest-Side Interface
Trust Domain
Unmodified Applications
Unmodified Drivers
TDX-Enlightened OS
Intel TDX Guest-Side Interface
TDX-Aware Host VMM
Intel TDX Host-Side Interface
Intel TDX Module Running in SEAM Root Mode
Platform (Cores, Caches, Devices etc.)

Intel TDX is a CC technology that provides confidentiality and integrity for data in use by tenant VMs, called Trust Domains (TDs)

- **Objective:** Remove the Virtual Machine Monitor (VMM) and other system SW from the TDs' TCB
- **TDX Module:** Intel-signed security services module responsible for enforcing security policies for TDs
- **SEAM:** A new Secure Arbitration Mode (SEAM) hosts the TDX module

TDX Threat Model

TRUSTED BY TD

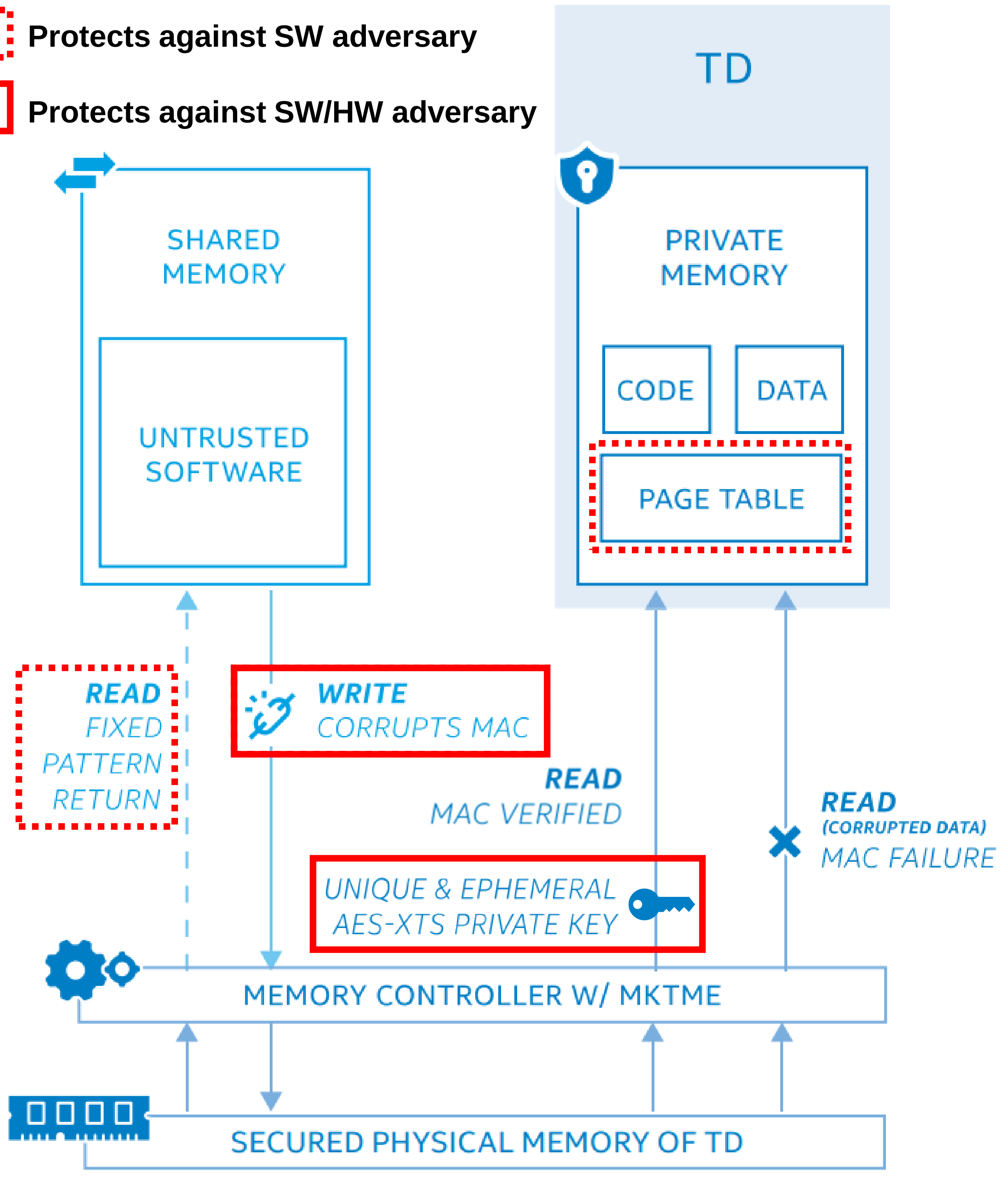
- INTEL® TDX MODULE
- INTEL AUTHENTHICATED CODE MODULES (ACM)
- TD QUOTING ENCLAVE
- INTEL CPU HARDWARE

NOT TRUSTED BY TD

- PLATFORM ADMIN
- DISCRETE AND INTEGRATED DEVICES
- ALL OTHER SOFTWARE
- OTHER PLATFORM FIRMWARE
- HOST-OS/VMM
- BIOS/SMM

Some side-channel attacks are out of scope for the TDX threat model

⬚	**Protects against SW adversary**
▭	**Protects against SW/HW adversary**


TD
SHARED MEMORY
UNTRUSTED SOFTWARE
PRIVATE MEMORY
CODE
DATA
PAGE TABLE
READ FIXED PATTERN RETURN
WRITE CORRUPTS MAC
READ MAC VERIFIED
READ (CORRUPTED DATA) MAC FAILURE
UNIQUE & EPHEMERAL AES-XTS PRIVATE KEY
MEMORY CONTROLLER W/ MKTME
SECURED PHYSICAL MEMORY OF TD

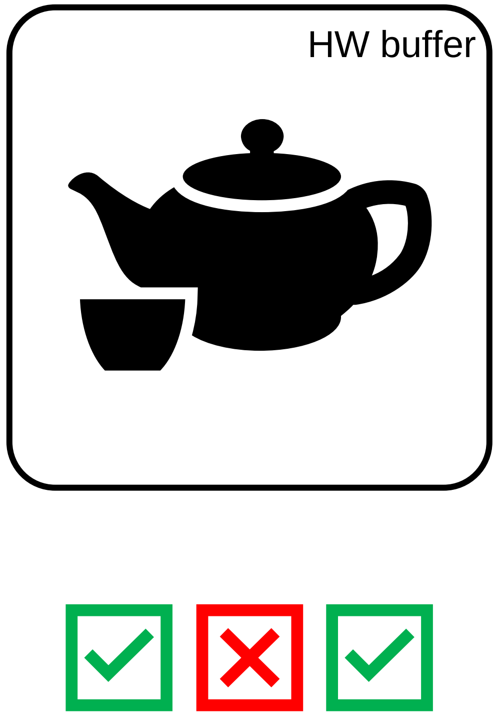
Source: White Paper | Intel® Trust Domain Extensions

What is a Side Channel?

Victim Context

Do a cat thing!
Do something else!
Do a different cat thing!

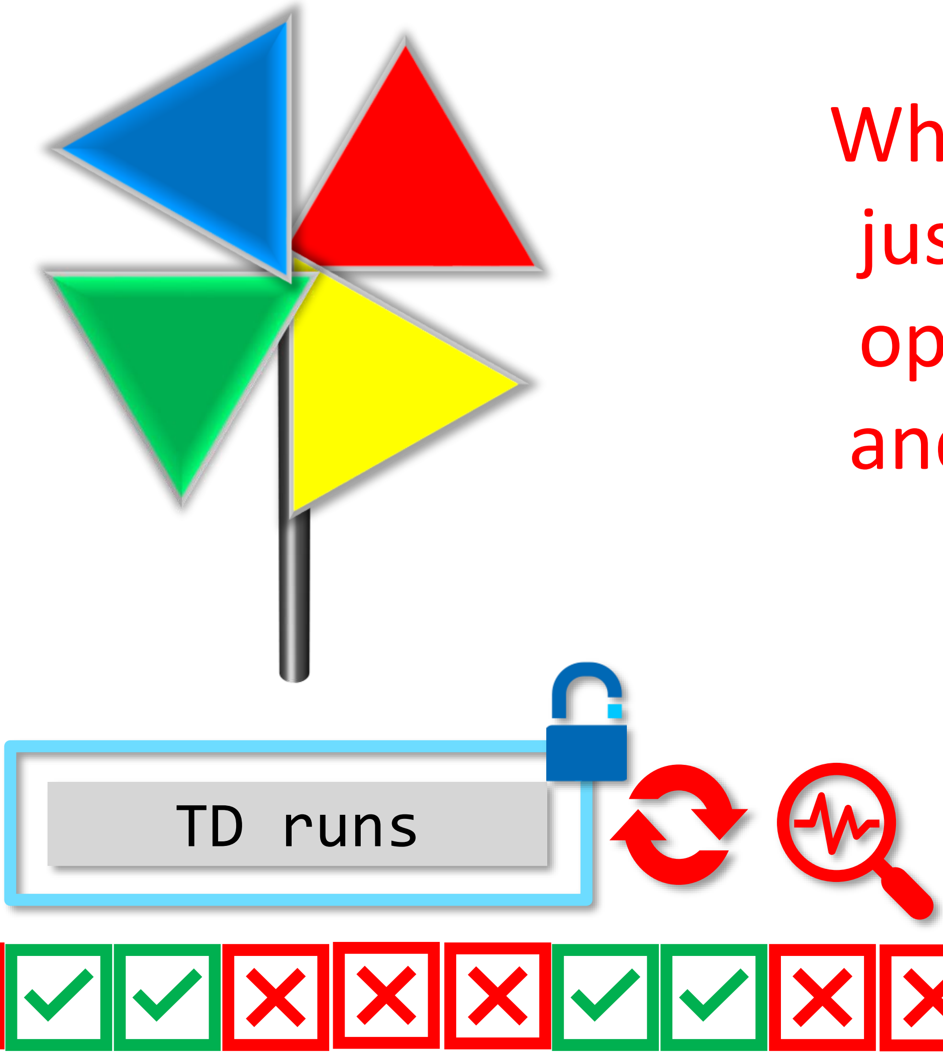
Attacker Context



✓ ✗ ✓

Do a teapot thing!
My teapot thing was *slow*, so the victim must have done a cat thing!
Do a teapot thing!
My teapot thing was *fast*, so the victim must *not* have done a cat thing!
Do a teapot thing!
My teapot thing was *slow*, so the victim must have done a cat thing!

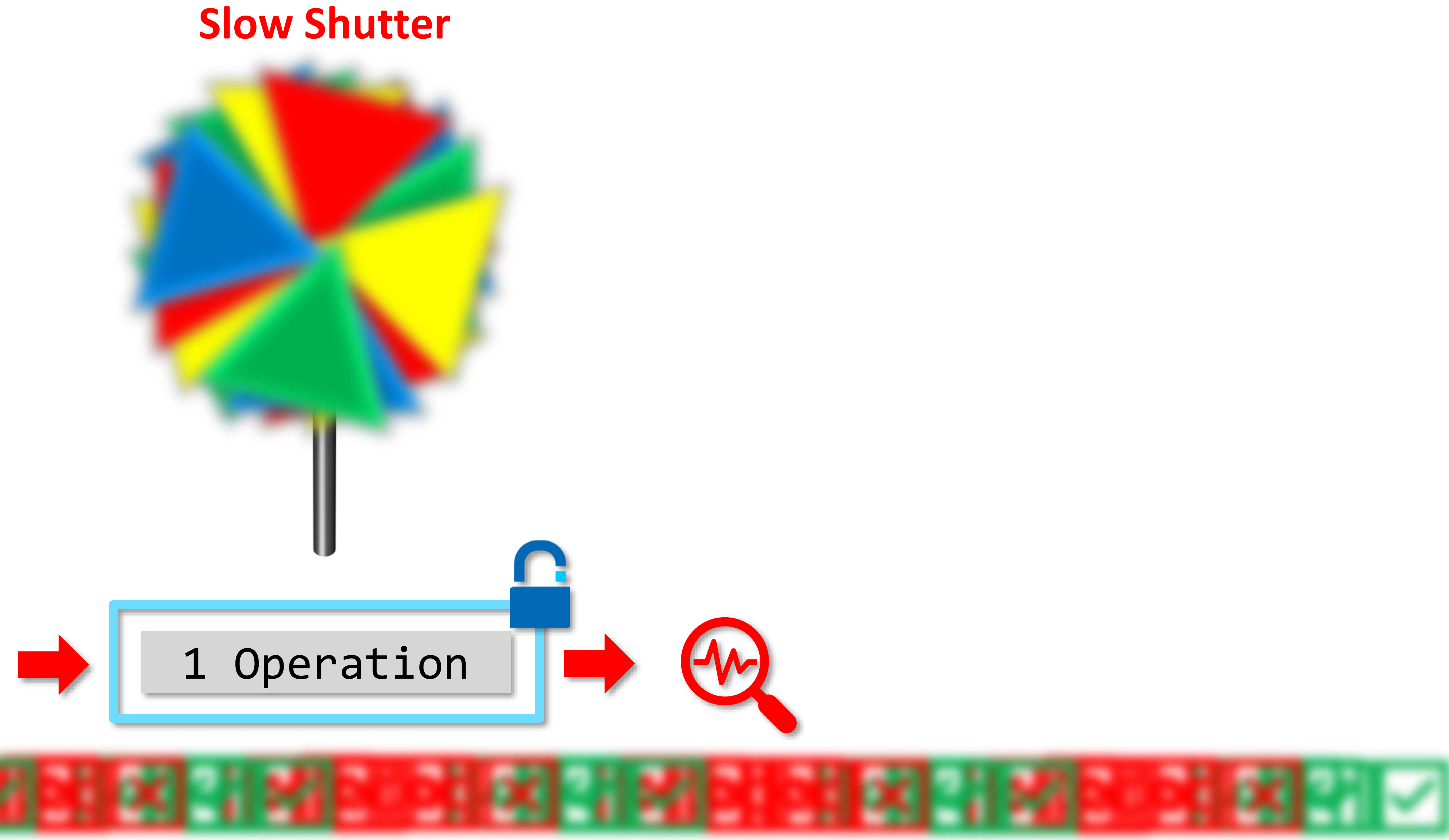
Adversary's Challenge: Shutter Speed



What if the adversary just triggers a single operation at a time, and then performs a side-channel measurement?

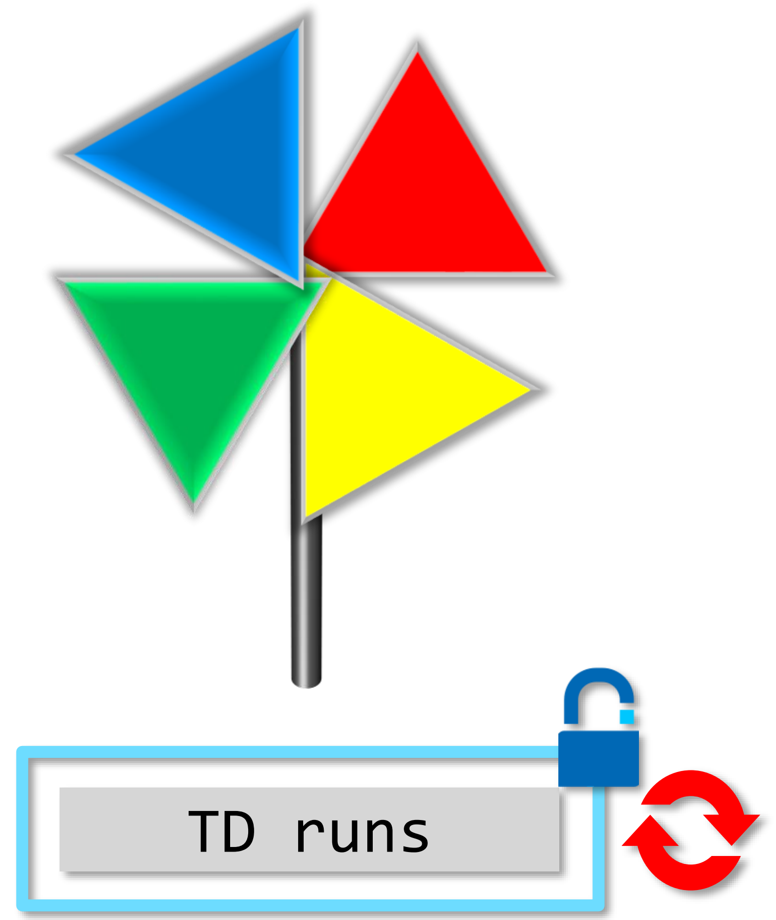
TD instruction stream
(typically 1.8 GHz or more):

Adversary's Challenge: Shutter Speed



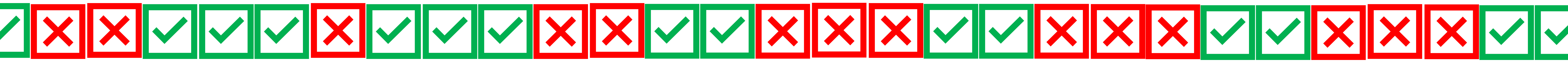
TD instruction stream
(typically 1.8 GHz or more):

Adversary’s Challenge: Shutter Speed

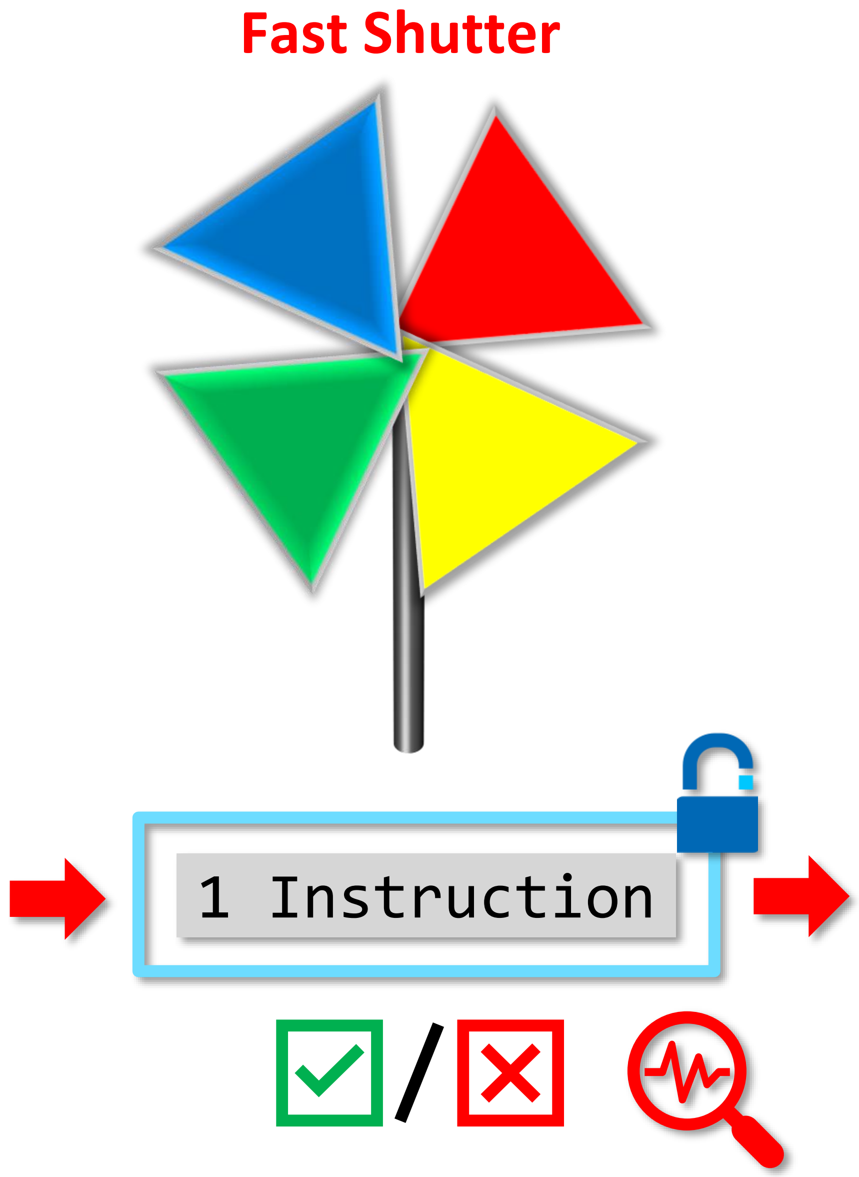


Let’s try something more precise: can the adversary trigger a single ***instruction*** at a time?

TD instruction stream
(typically 1.8 GHz or more):



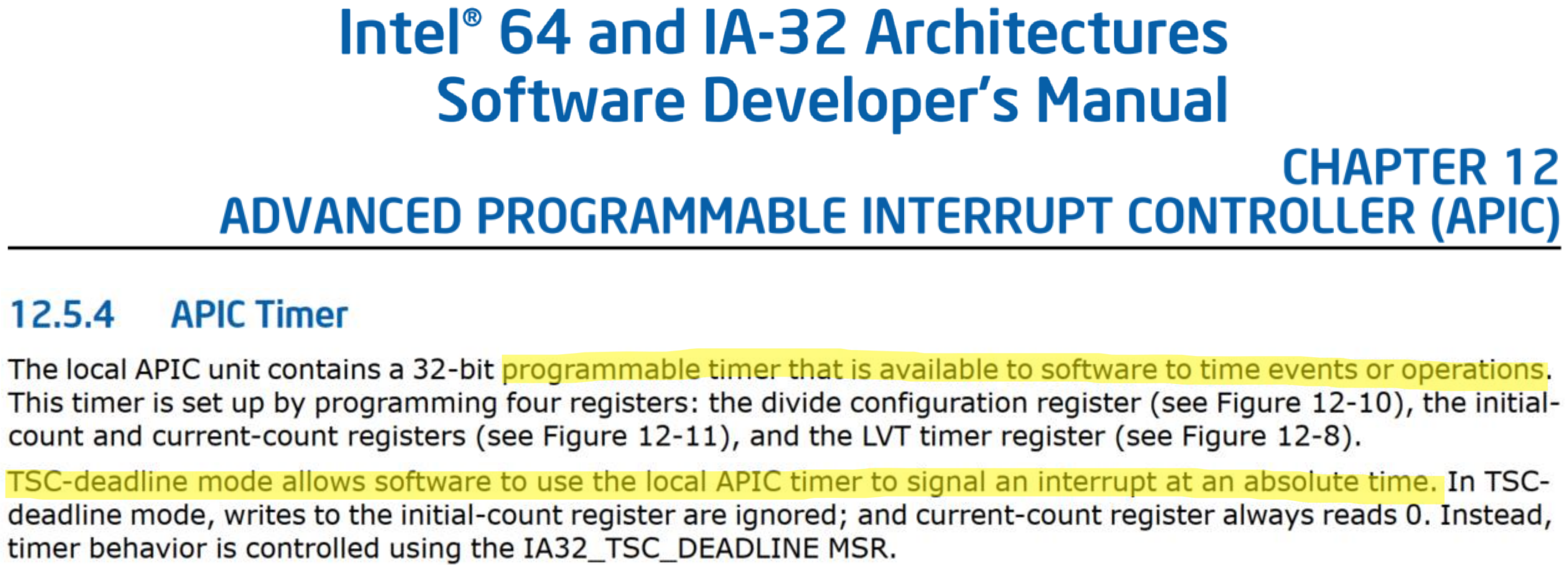
Adversary's Challenge: Shutter Speed



This technique was pioneered in a framework called:

SGX-Step

Searching for an Ideal HW Primitive



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

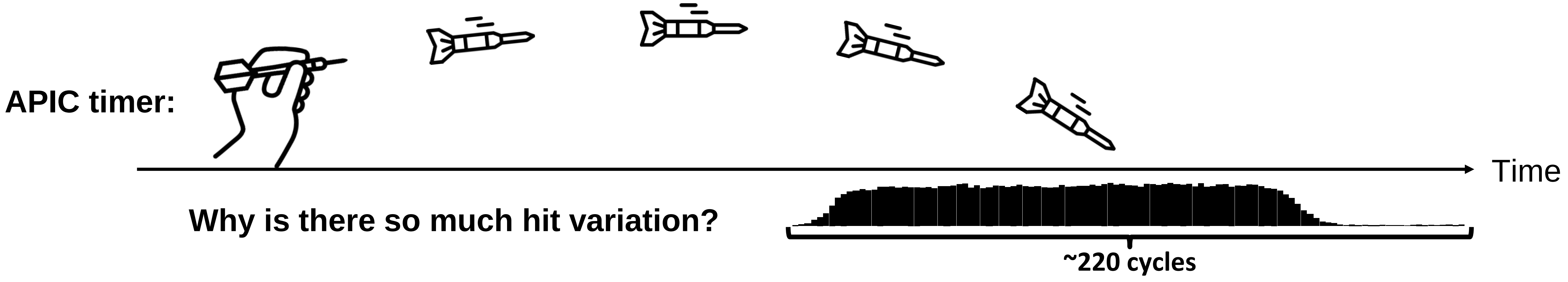
CHAPTER 12
ADVANCED PROGRAMMABLE INTERRUPT CONTROLLER (APIC)

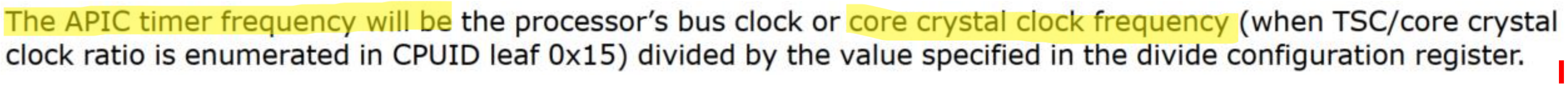
12.5.4 APIC Timer

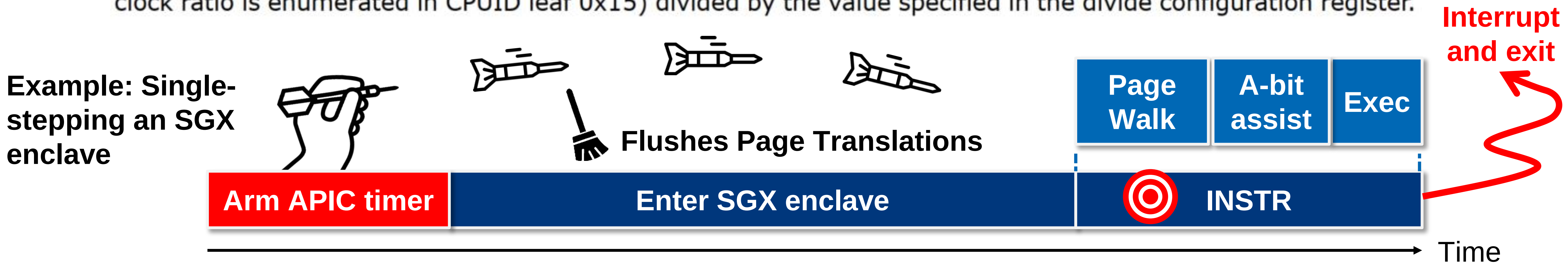
The local APIC unit contains a 32-bit programmable timer that is available to software to time events or operations. This timer is set up by programming four registers: the divide configuration register (see Figure 12-10), the initial-count and current-count registers (see Figure 12-11), and the LVT timer register (see Figure 12-8).

TSC-deadline mode allows software to use the local APIC timer to signal an interrupt at an absolute time. In TSC-deadline mode, writes to the initial-count register are ignored; and current-count register always reads 0. Instead, timer behavior is controlled using the IA32_TSC_DEADLINE MSR.

A Small Target for an (Inaccurate) APIC



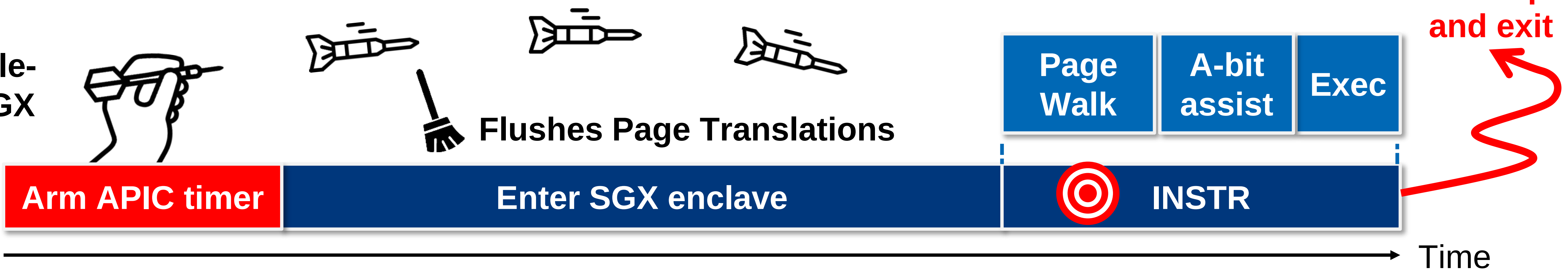

The APIC timer frequency will be the processor's bus clock or core crystal clock frequency (when TSC/core crystal clock ratio is enumerated in CPUID leaf 0x15) divided by the value specified in the divide configuration register.



A Small Target for an (Inaccurate) APIC

Key Idea 1: Don't try to hit a small target!
Instead, make the target *slower* and therefore
bigger

Example: Single-stepping an SGX enclave

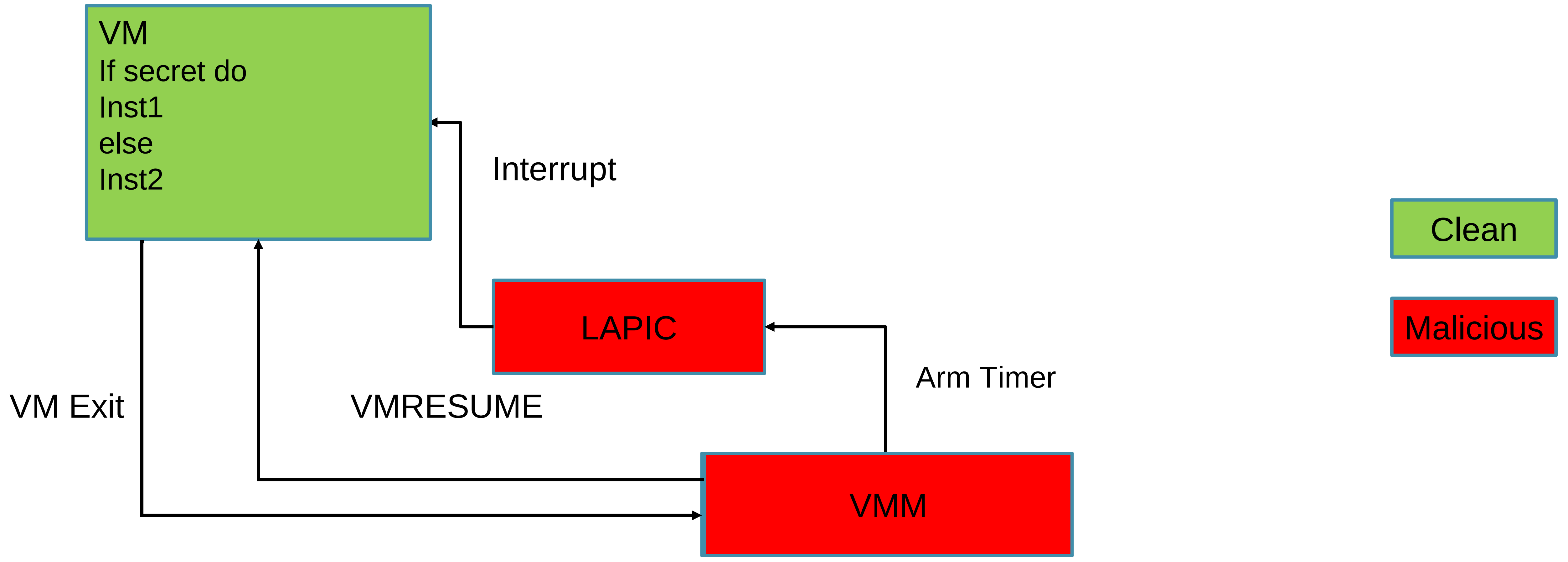


Agenda

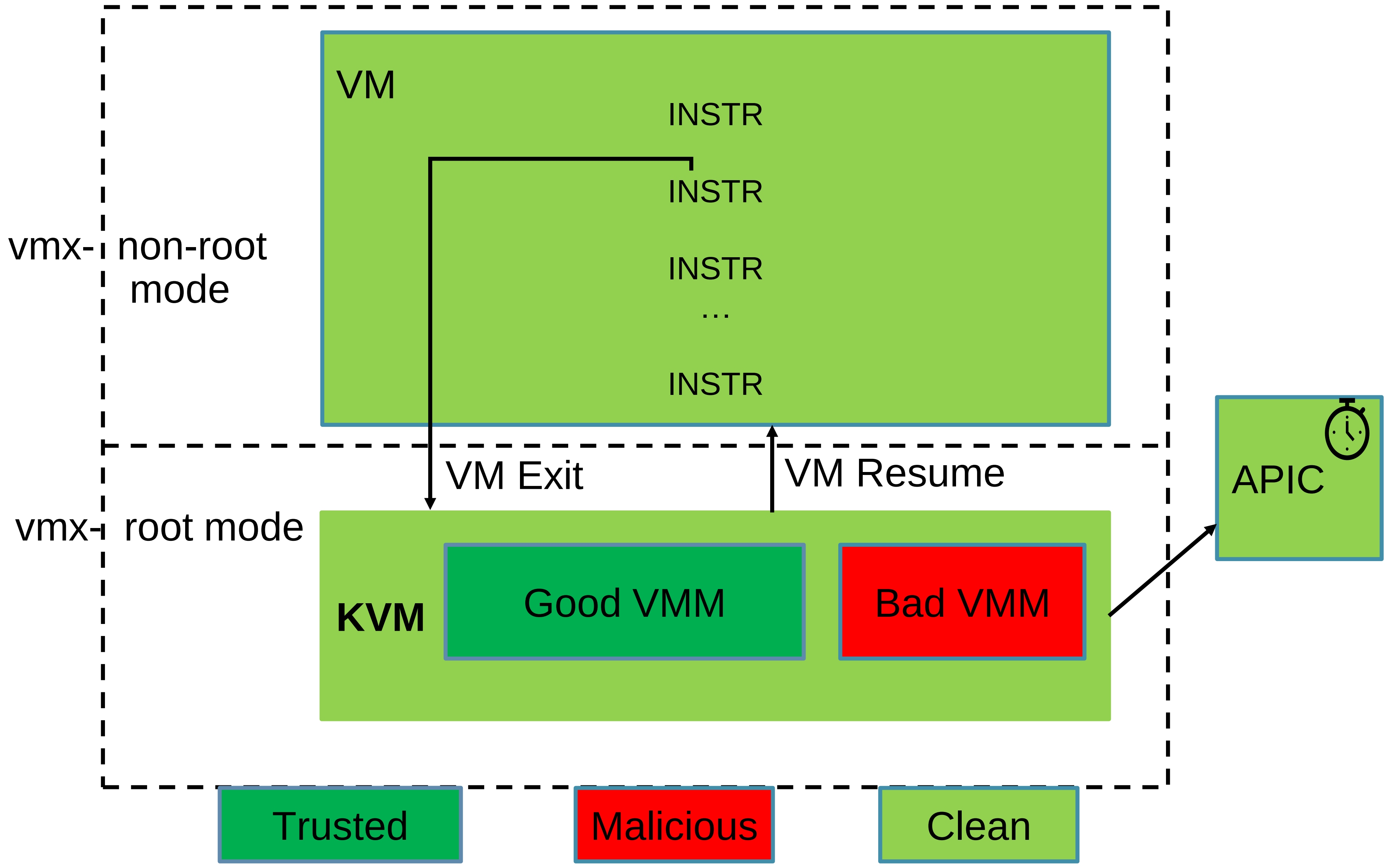
- Intro to Confidential Computing, Intel TDX (Trust Domain Extensions), side-channel Attacks, and malicious single-stepping
- Pre-TDX PoC (Proof of Concept) TDX-step exploit and mitigation
- Techniques to bypass the TDX-Step mitigation, and intro to the new ICSSD (Instruction Counting Single-Step Defense) feature
- Comparison with the SGX-Step mitigation

TDX-Step Attack Scenario

- **PoC (Proof of Concept) Exploit** was developed in a virtualization setup (no TDX support)
- **Interrupt based attack:** VMM mounts attack on VM/TD application using APIC

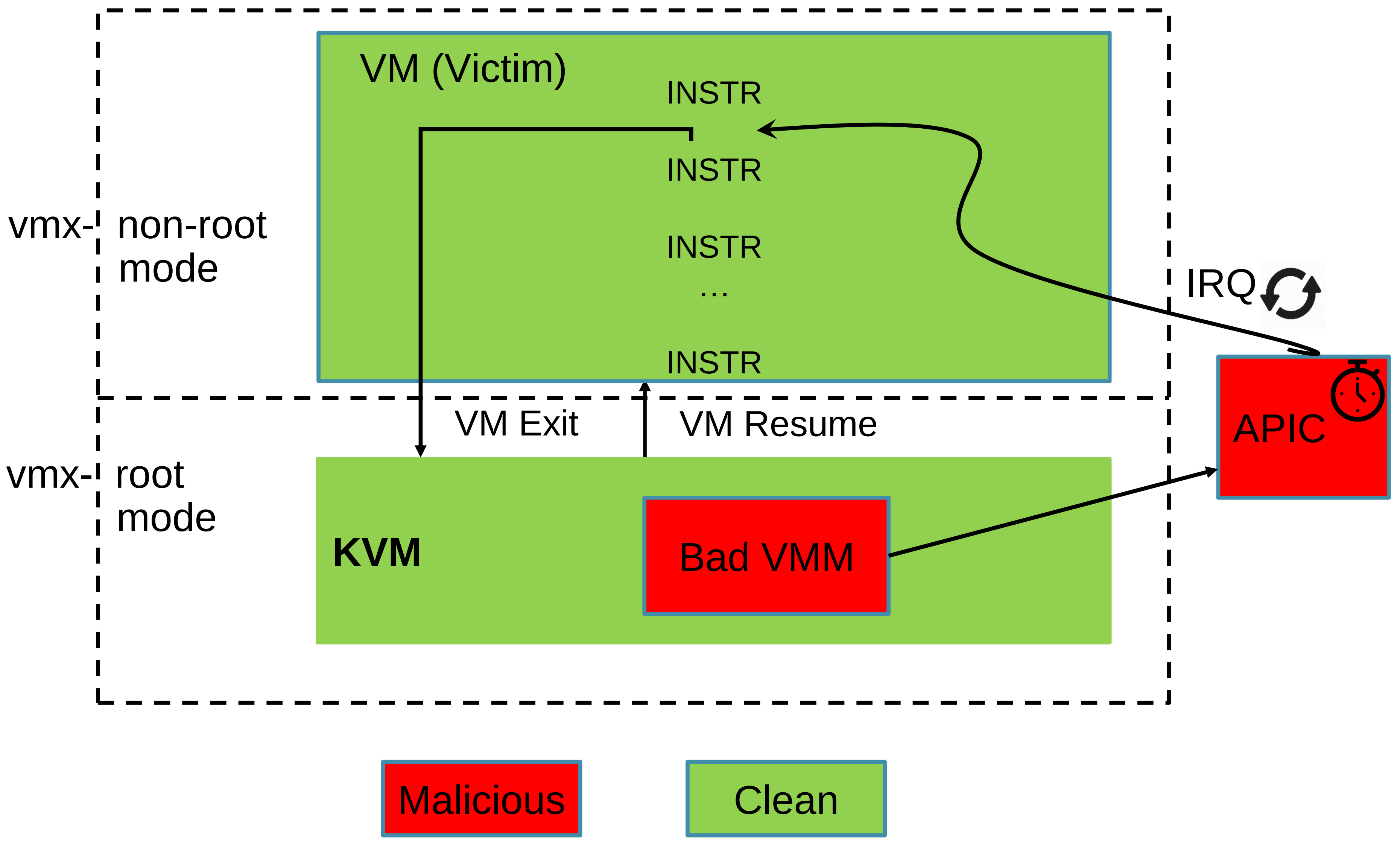


Pre-TDX Test Setup



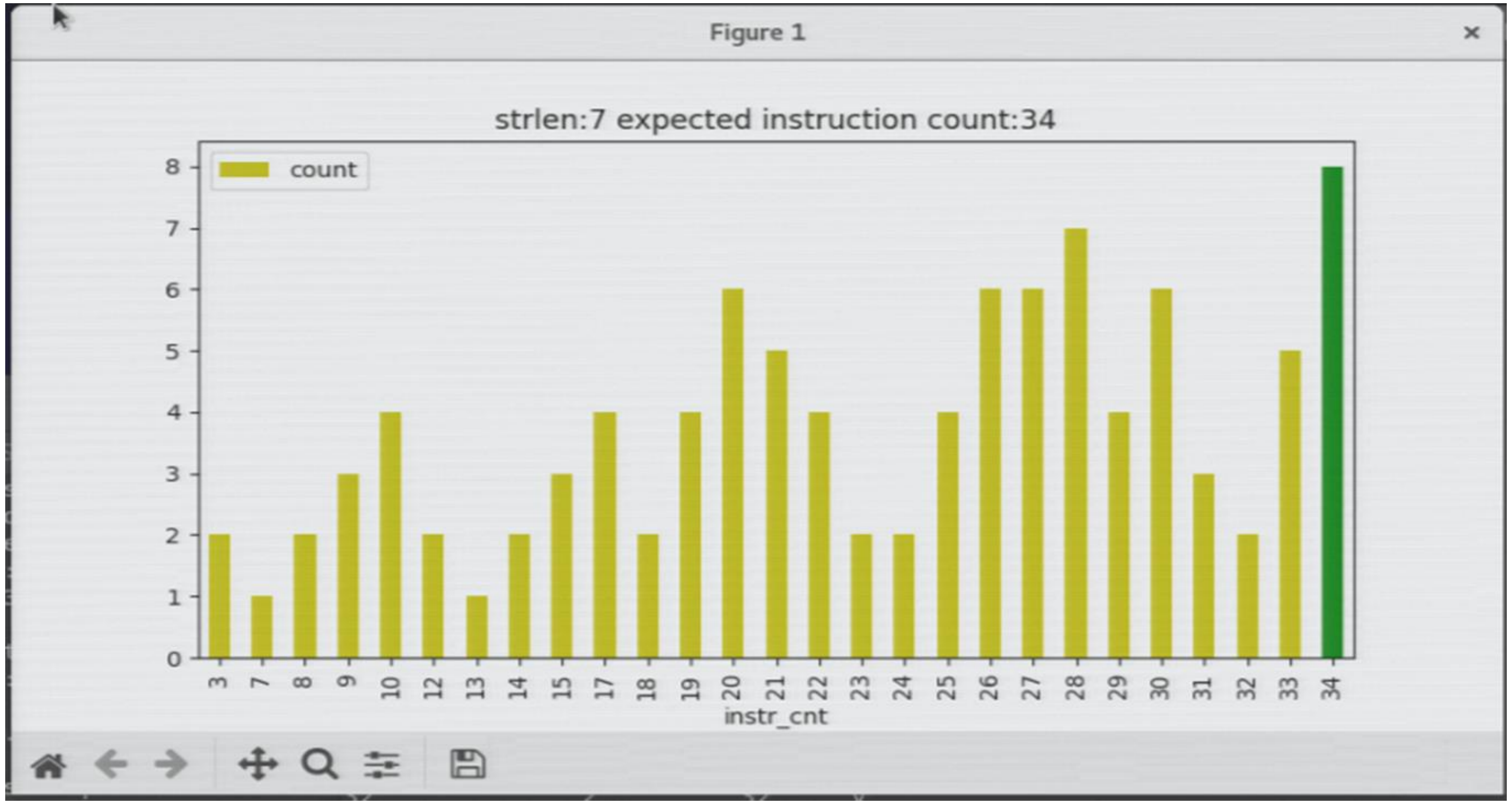
- KVM (Host VMM) as the host used as the both Good (Mitigation) and Bad (Attack) Actor
- Ubuntu VM (Virtual Machine) as the Guest, running “strlen” workload/app as a victim app
- Good VMM portion of the KVM code was eventually implemented in Intel TDX Module, which is responsible for mitigating the TDX-Step attack

TDX-Step Attack Realized



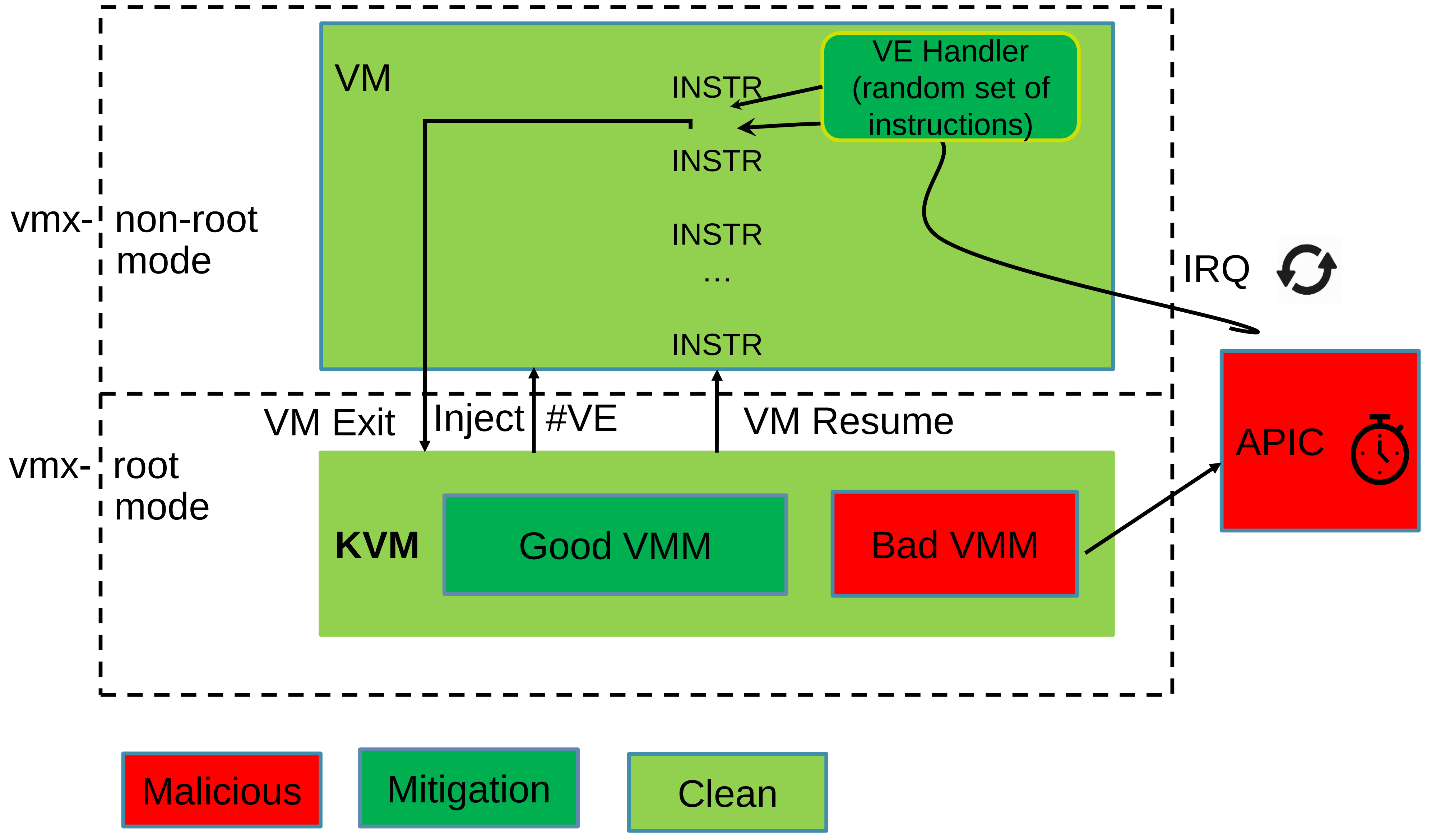
- Bad VMM configures the APIC timer to cause periodic interrupts to Victim VM on an instruction boundary
- Once interrupted, victim VM will exit to the host through VM Exit
- Now Bad VMM has the ability conduct side channel analysis on the VM instruction execution
- Also, bad VMM will cause VM Resume to make the victim VM to execute all the instructions

Proof of Concept attack: Interrupt-based attack on strlen



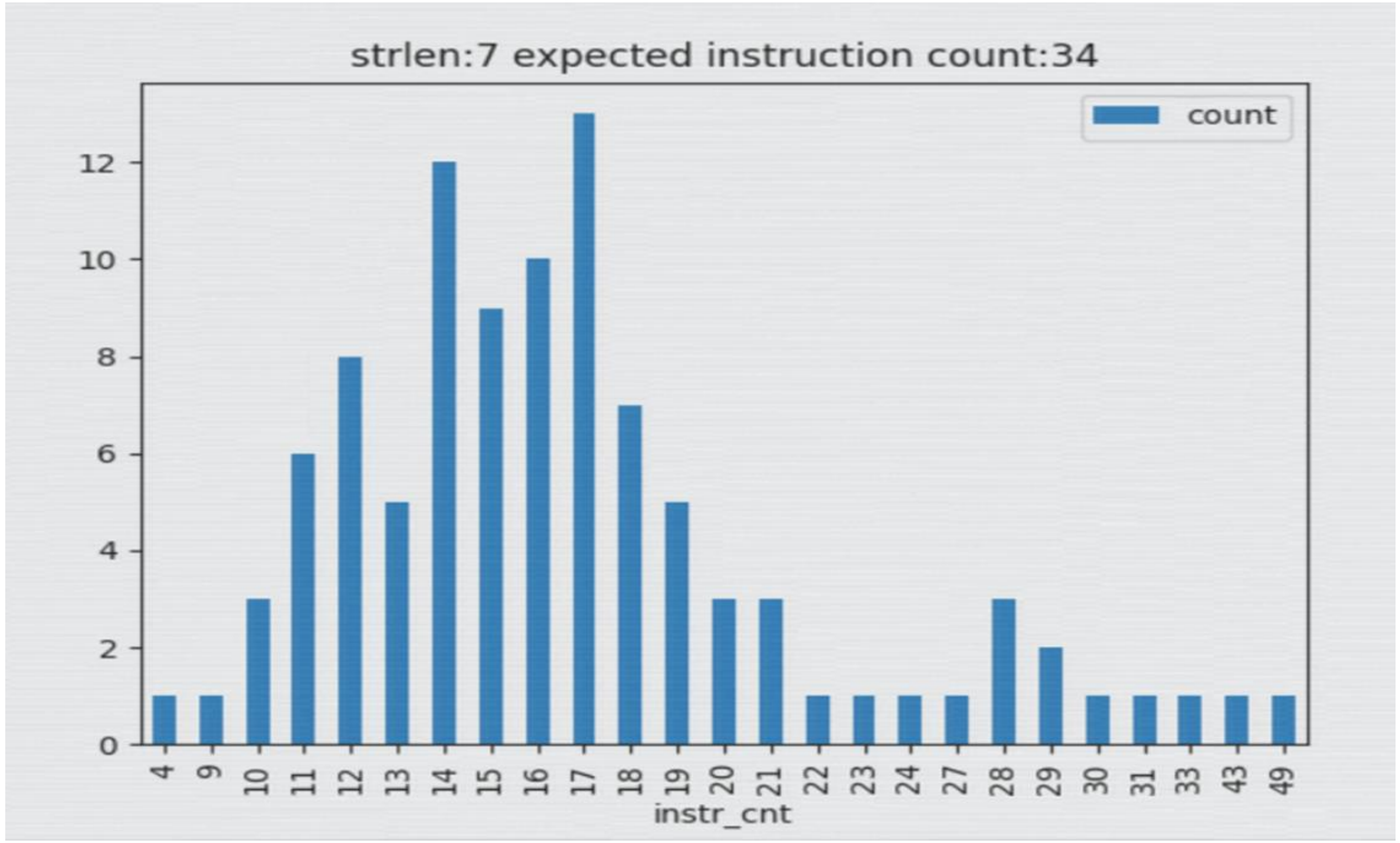
- Victim VM executing the strlen 7 consisted of 34 instructions
- Out of 100 experiments, the PoC attack was able to cause single-step attack successfully for 8 times
- This was sufficient to prove that TDX-Step attack indeed is real

Mitigation - Proposal 1



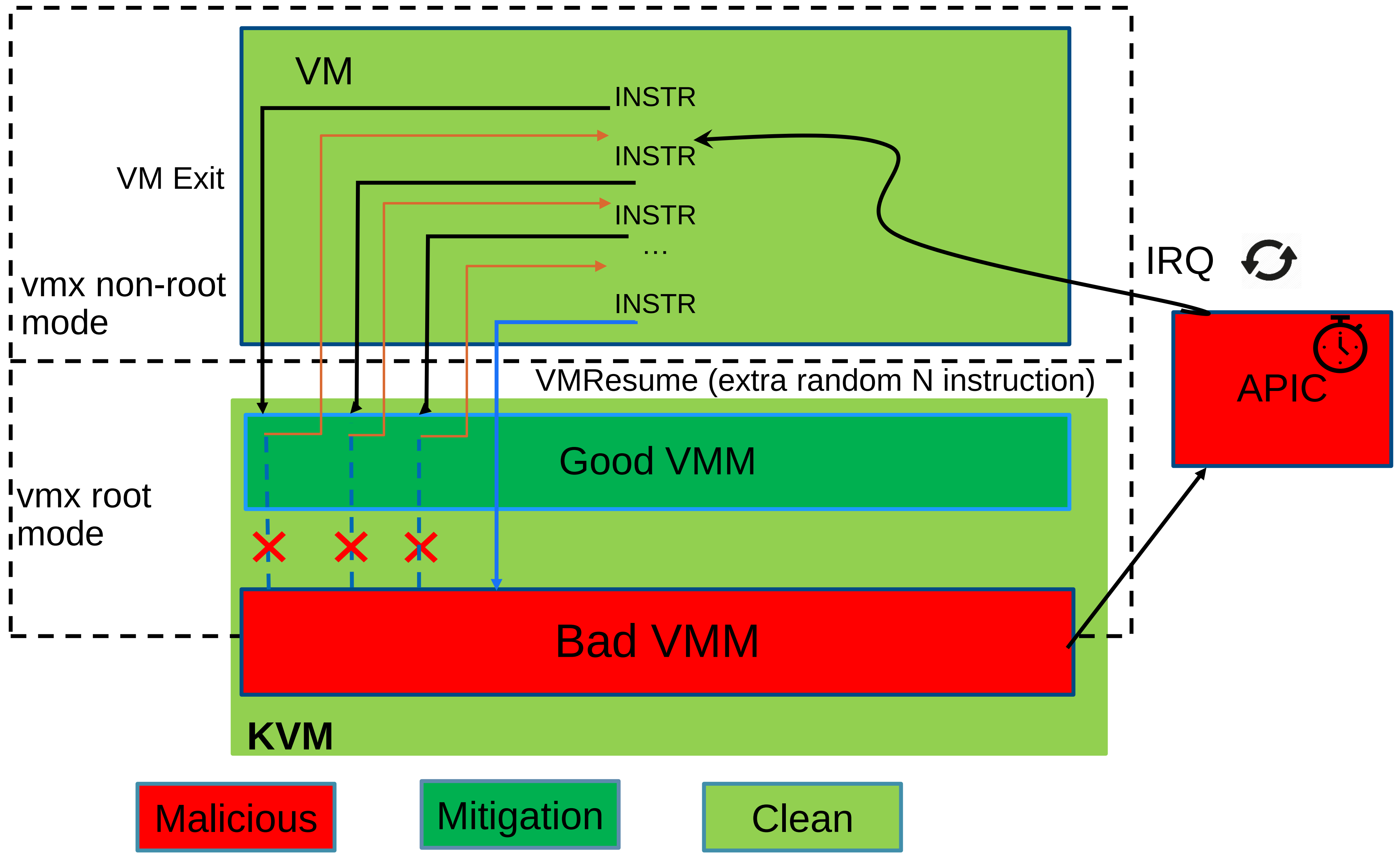
- **Mitigation:** “Good VMM” portion of KVM implements the step-detector which will inject #VE (Virtualization Exception) to VM
- VM will eventually exit to the Host

Proposal 1: Mitigation PoC Result



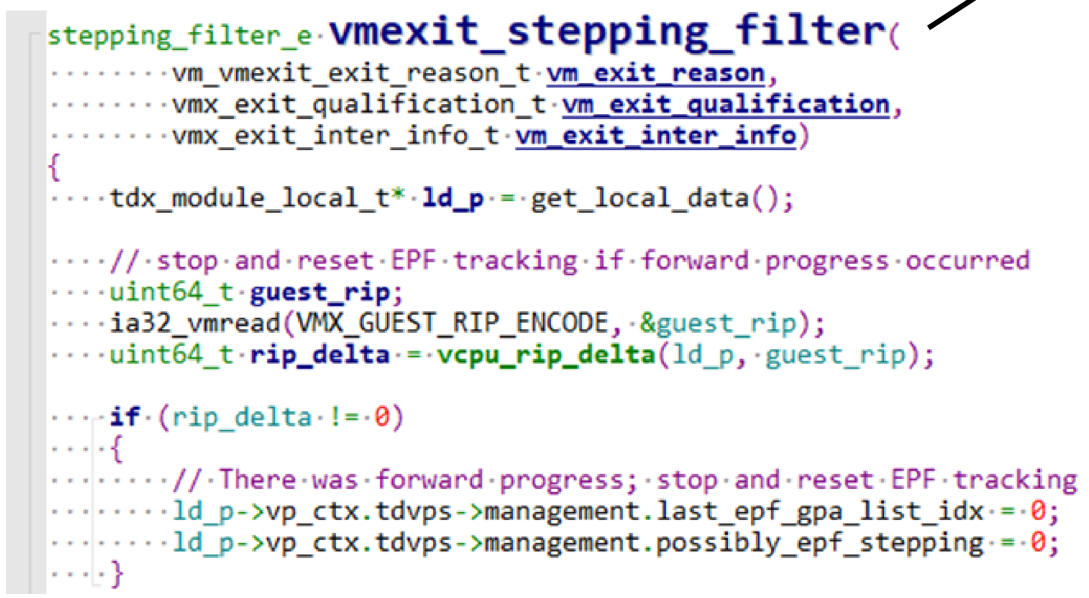
- Test results show that it wasn't possible to cause VM Exits exactly on every instruction boundary (34 instructions)
- Even if so, the real instruction count of 34 was never met
- **Limitations**
  1) NOT scalable
  2) Increased attack surface
- **Next step:** Define a mitigation to overcome the above limitations

Mitigation - Proposal 2



- Mitigation is fully self-contained within the “Good VMM”, employing the step-filter
- Resumes the victim VM to execute extra “N” random instructions before it transfer the control back to the Bad VMM
- **Result:** With the mitigation enabled, instruction count never matched the expected count value of 34, thereby defeating the attack

TDX-Step (v1.0) Mitigation using TDX Module



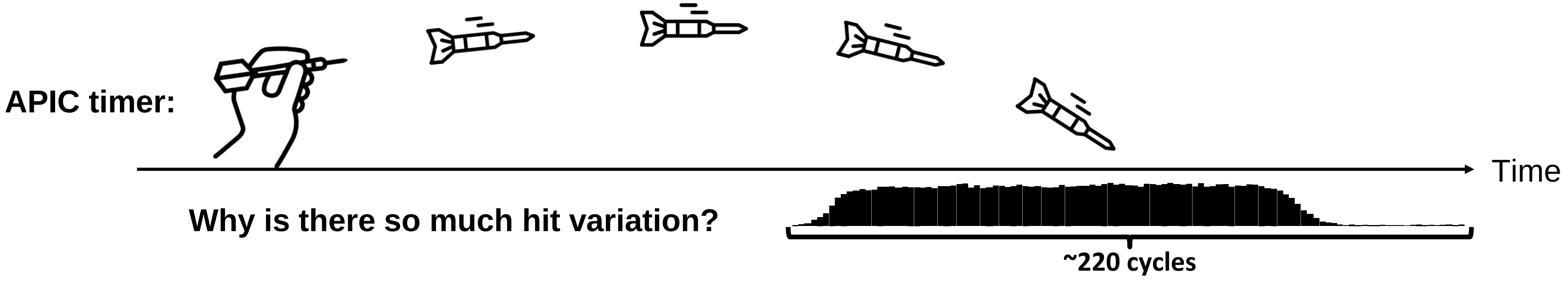
```
stepping_filter_e vmexit_stepping_filter(
        vm_vmexit_exit_reason_t vm_exit_reason,
        vmx_exit_qualification_t vm_exit_qualification,
        vmx_exit_inter_info_t vm_exit_inter_info)
{
    tdx_module_local_t* ld_p = get_local_data();

    // stop and reset EPF tracking if forward progress occurred
    uint64_t guest_rip;
    ia32_vmread(VMX_GUEST_RIP_ENCODE, &guest_rip);
    uint64_t rip_delta = vcpu_rip_delta(ld_p, guest_rip);

    if (rip_delta != 0)
    {
        // There was forward progress; stop and reset EPF tracking
        ld_p->vp_ctx.tdvps->management.last_epf_gpa_list_idx = 0;
        ld_p->vp_ctx.tdvps->management.possibly_epf_stepping = 0;
    }
```

- TDX Module @ **https://github.com/intel/tdx-module**
- The TDX module employs a VM exit Step-filter algorithm
- Pre-conditions to enable step filter:
  1. Interrupt duration should be less 4K cycles since the last entry –**TSC (Time Stamp Counter)**) **or**
  2. The RIP of the VCPU has not made progress
- TDX Module resumes the VCPU in “stepping mode” for random number (2-32) of instructions
- VCPU stepping is done using the VMX Monitor Trap Flag (MTF) mechanism.

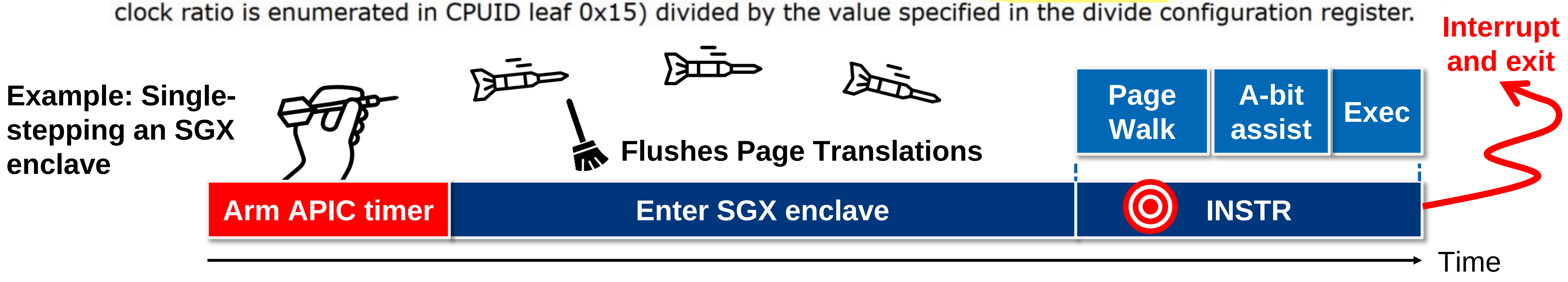
Agenda

- Intro to Confidential Computing, Intel TDX (Trust Domain Extensions), side-channel Attacks, and malicious single-stepping
- Pre-TDX PoC (Proof of Concept) TDX-step exploit and mitigation
- Techniques to bypass the TDX-Step mitigation, and intro to the new ICSSD (Instruction Counting Single-Step Defense) feature
- Comparison with the SGX-Step mitigation

(We showed this slide earlier)



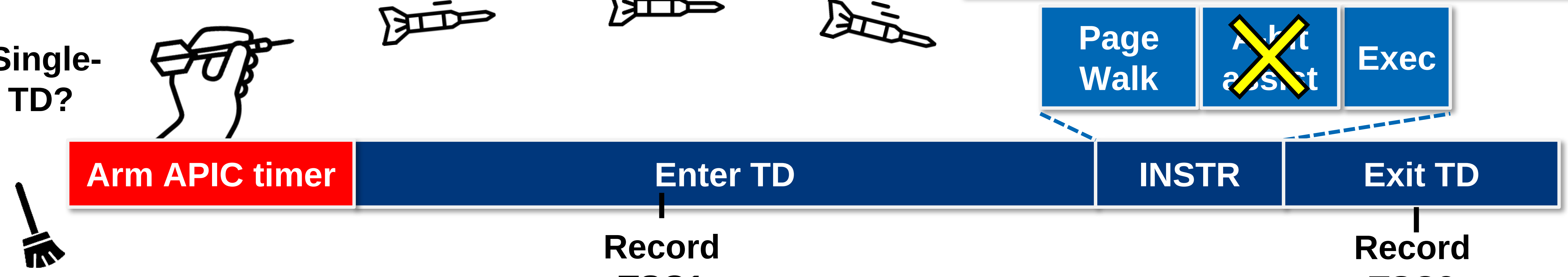
The APIC timer frequency will be the processor's bus clock or core crystal clock frequency (when TSC/core crystal clock ratio is enumerated in CPUID leaf 0x15) divided by the value specified in the divide configuration register.



Attempting to Single-step a TD

Example: Single-stepping a TD?

Recall that each TD's page tables are protected within its own private memory



Record TSC1

Record TSC2

Flushes Page Translations

Recall that the TD forces RAND[2,32] instructions to execute if less than 4K cycles elapse between entry and exit

If (TSC2-TSC1 < 4K): Apply mitigation

Demo 1

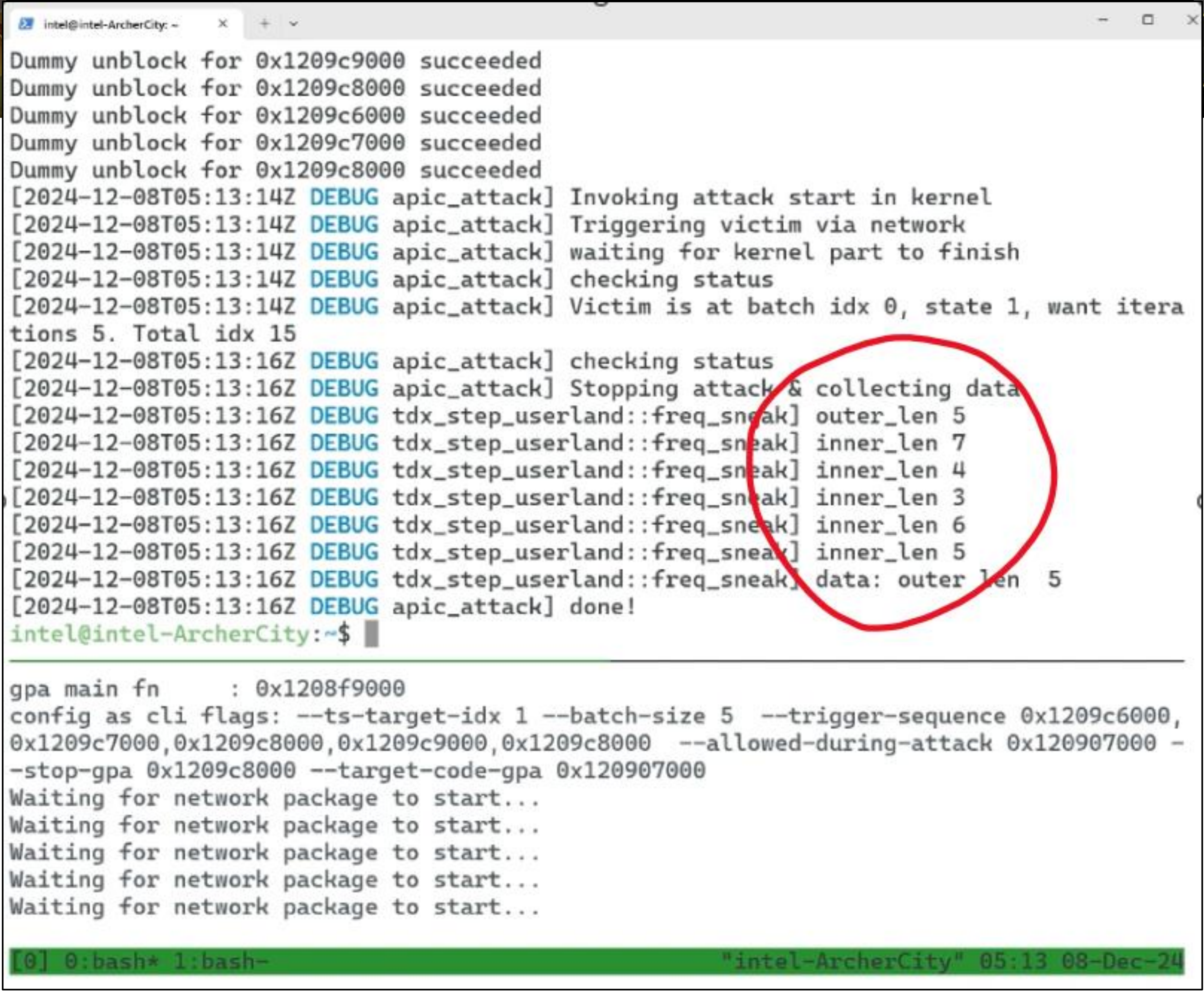
- Let's attempt to single-step a TD
- (Note that RAX=10 at the start)

Listing 1: Evaluation target for single-stepping.

```
  mov   qword ptr[r8], 42
loop_label:
  dec  rax
  nop
  nop
  nop
  nop
  jnz  loop_label
  mov  qword ptr [r9], 42
```

62 instructions
÷
~16 instructions forced by the TDX module
=
~4 instructions per attack attempt

Listing Source: L. Wilke, F. Sieck, and T. Eisenbarth, 'TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX', in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14--18, 2024*, 2024.

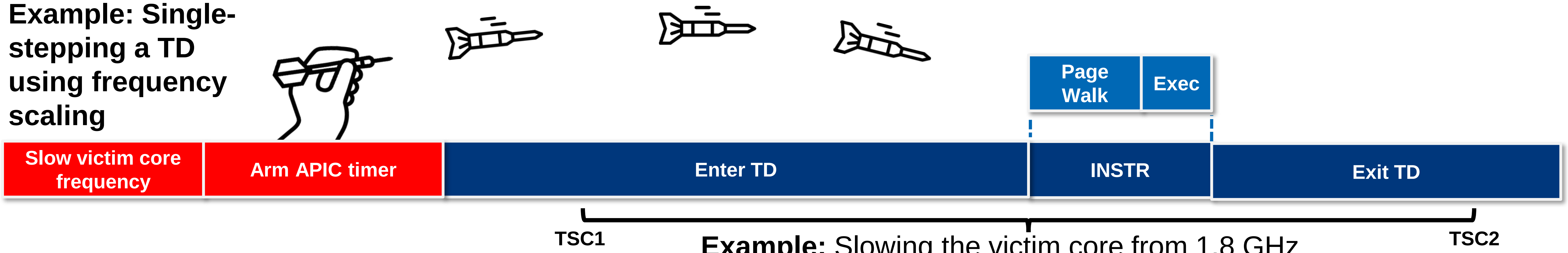


```
Dummy unblock for 0x1209c9000 succeeded
Dummy unblock for 0x1209c8000 succeeded
Dummy unblock for 0x1209c6000 succeeded
Dummy unblock for 0x1209c7000 succeeded
Dummy unblock for 0x1209c8000 succeeded
[2024-12-08T05:13:14Z DEBUG apic_attack] Invoking attack start in kernel
[2024-12-08T05:13:14Z DEBUG apic_attack] Triggering victim via network
[2024-12-08T05:13:14Z DEBUG apic_attack] waiting for kernel part to finish
[2024-12-08T05:13:14Z DEBUG apic_attack] checking status
[2024-12-08T05:13:14Z DEBUG apic_attack] Victim is at batch idx 0, state 1, want itera
tions 5. Total idx 15
[2024-12-08T05:13:16Z DEBUG apic_attack] checking status
[2024-12-08T05:13:16Z DEBUG apic_attack] Stopping attack & collecting data
[2024-12-08T05:13:16Z DEBUG tdx_step_userland::freq_sneak] outer_len 5
[2024-12-08T05:13:16Z DEBUG tdx_step_userland::freq_sneak] inner_len 7
[2024-12-08T05:13:16Z DEBUG tdx_step_userland::freq_sneak] inner_len 4
[2024-12-08T05:13:16Z DEBUG tdx_step_userland::freq_sneak] inner_len 3
[2024-12-08T05:13:16Z DEBUG tdx_step_userland::freq_sneak] inner_len 6
[2024-12-08T05:13:16Z DEBUG tdx_step_userland::freq_sneak] inner_len 5
[2024-12-08T05:13:16Z DEBUG tdx_step_userland::freq_sneak] data: outer len   5
[2024-12-08T05:13:16Z DEBUG apic_attack] done!
intel@intel-ArcherCity:~$ 

gpa main fn      : 0x1208f9000
config as cli flags: --ts-target-idx 1 --batch-size 5  --trigger-sequence 0x1209c6000,
0x1209c7000,0x1209c8000,0x1209c9000,0x1209c8000  --allowed-during-attack 0x120907000 -
-stop-gpa 0x1209c8000 --target-code-gpa 0x120907000
Waiting for network package to start...
Waiting for network package to start...
Waiting for network package to start...
Waiting for network package to start...
Waiting for network package to start...

[0] 0:bash* 1:bash-                              "intel-ArcherCity" 05:13 08-Dec-24
```

Information Classification: General

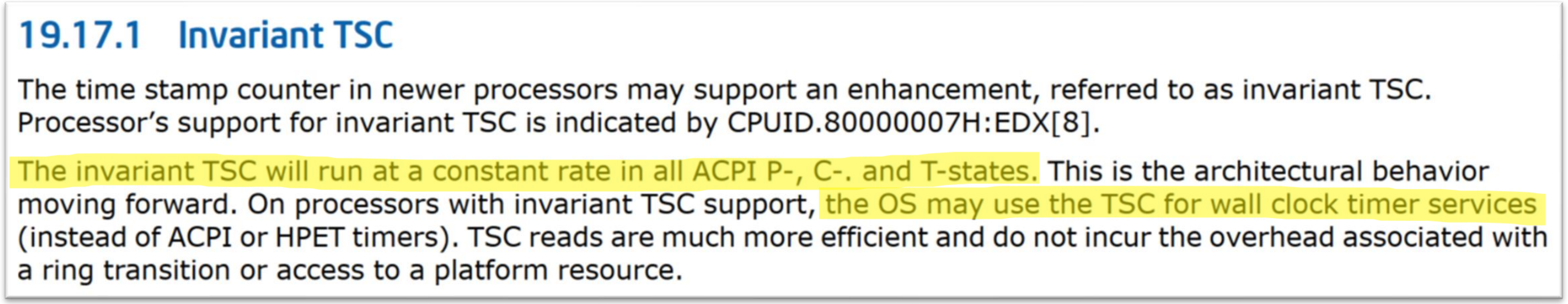
Single-stepping a TD (TDXdown)

Example: Single-stepping a TD using frequency scaling



Example: Slowing the victim core from 1.8 GHz to 800 MHz will increase the latency of all victim operations by 2.25x, as measured by the TSC

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



19.17.1 Invariant TSC

The time stamp counter in newer processors may support an enhancement, referred to as invariant TSC. Processor's support for invariant TSC is indicated by CPUID.80000007H:EDX[8].

The invariant TSC will run at a constant rate in all ACPI P-, C-. and T-states. This is the architectural behavior moving forward. On processors with invariant TSC support, the OS may use the TSC for wall clock timer services (instead of ACPI or HPET timers). TSC reads are much more efficient and do not incur the overhead associated with a ring transition or access to a platform resource.

Demo 2

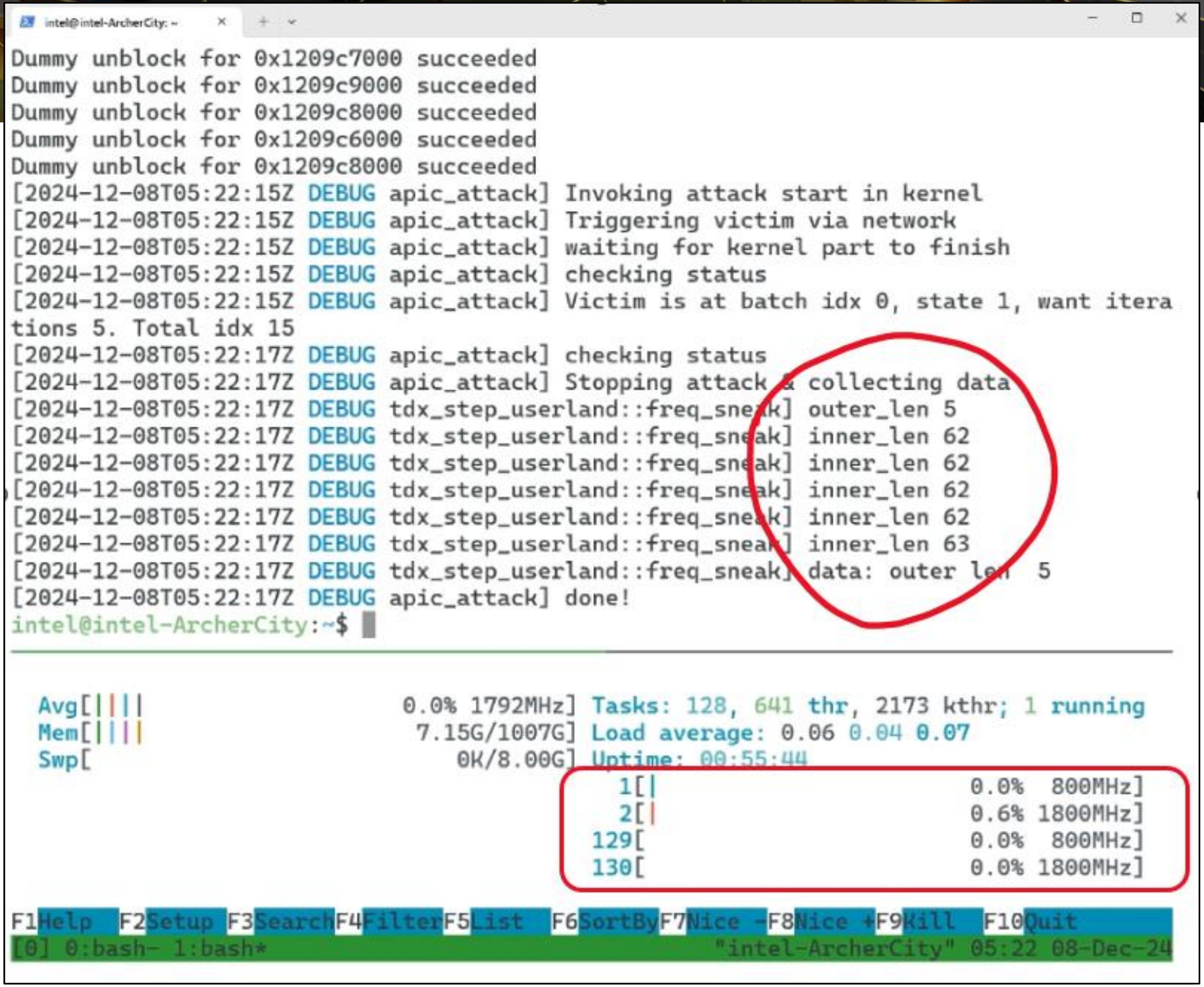
- Let’s attempt to single-step a TD using frequency scaling (TDXdown technique)

Listing 1: Evaluation target for single-stepping.

```
  mov  qword ptr[r8], 42
loop_label:
  dec rax
  nop
  nop
  nop
  nop
  jnz loop_label
  mov qword ptr [r9], 42
```

62 instructions

Listing Source: L. Wilke, F. Sieck, and T. Eisenbarth, ‘TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX’, in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14--18, 2024*, 2024.

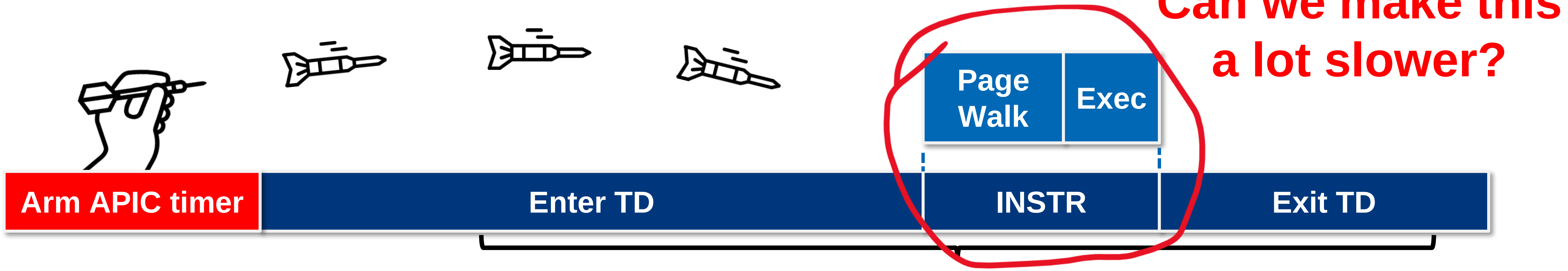


intel@intel-ArcherCity: ~

```
Dummy unblock for 0x1209c7000 succeeded
Dummy unblock for 0x1209c9000 succeeded
Dummy unblock for 0x1209c8000 succeeded
Dummy unblock for 0x1209c6000 succeeded
Dummy unblock for 0x1209c8000 succeeded
[2024-12-08T05:22:15Z DEBUG apic_attack] Invoking attack start in kernel
[2024-12-08T05:22:15Z DEBUG apic_attack] Triggering victim via network
[2024-12-08T05:22:15Z DEBUG apic_attack] waiting for kernel part to finish
[2024-12-08T05:22:15Z DEBUG apic_attack] checking status
[2024-12-08T05:22:15Z DEBUG apic_attack] Victim is at batch idx 0, state 1, want itera
tions 5. Total idx 15
[2024-12-08T05:22:17Z DEBUG apic_attack] checking status
[2024-12-08T05:22:17Z DEBUG apic_attack] Stopping attack & collecting data
[2024-12-08T05:22:17Z DEBUG tdx_step_userland::freq_sneak] outer_len 5
[2024-12-08T05:22:17Z DEBUG tdx_step_userland::freq_sneak] inner_len 62
[2024-12-08T05:22:17Z DEBUG tdx_step_userland::freq_sneak] inner_len 62
[2024-12-08T05:22:17Z DEBUG tdx_step_userland::freq_sneak] inner_len 62
[2024-12-08T05:22:17Z DEBUG tdx_step_userland::freq_sneak] inner_len 62
[2024-12-08T05:22:17Z DEBUG tdx_step_userland::freq_sneak] inner_len 63
[2024-12-08T05:22:17Z DEBUG tdx_step_userland::freq_sneak] data: outer len 5
[2024-12-08T05:22:17Z DEBUG apic_attack] done!
intel@intel-ArcherCity:~$ 

  Avg[||||                    0.0% 1792MHz] Tasks: 128, 641 thr, 2173 kthr; 1 running
  Mem[||||                  7.15G/1007G] Load average: 0.06 0.04 0.07
  Swp[                        0K/8.00G] Uptime: 00:55:44
                                   1[|                          0.0%  800MHz]
                                   2[|                          0.6% 1800MHz]
                                 129[                           0.0%  800MHz]
                                 130[                           0.0% 1800MHz]
F1Help  F2Setup F3SearchF4FilterF5List   F6SortByF7Nice -F8Nice +F9Kill   F10Quit
[0] 0:bash- 1:bash*                                 "intel-ArcherCity" 05:22 08-Dec-24
```

Key Idea 2: Frequency scaling can be used to fool mitigation heuristics that rely on the TSC

Single-stepping a TD (Method 2)



Slowing these operations is the focus of TDXdown

Key Idea 1: Don't try to hit a small target!
Instead, make the target *slower* and therefore
bigger

A Page Walk isn't always a Cake Walk

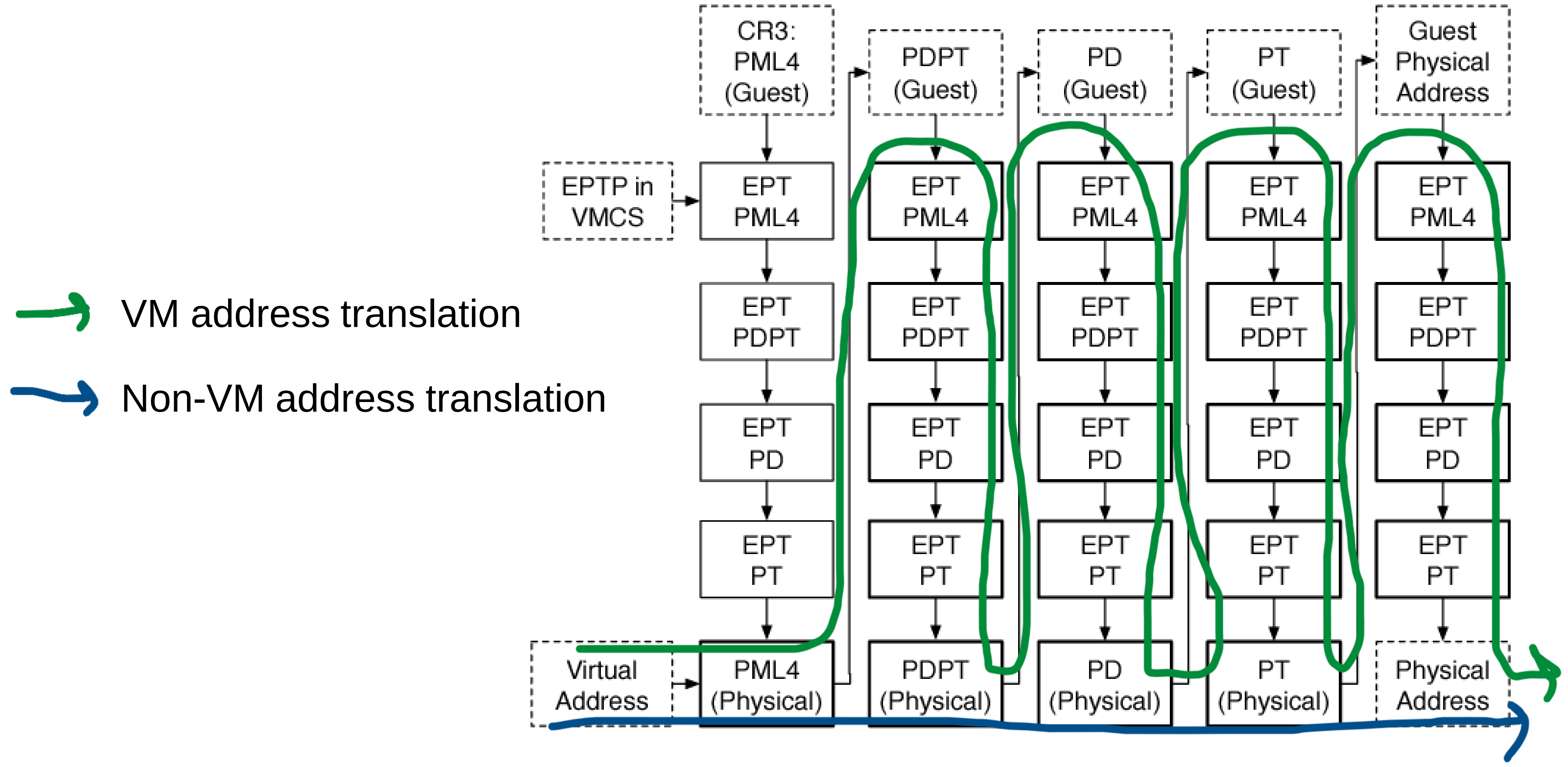


Image Source: https://rayanfam.com/topics/hypervisor-from-scratch-part-4/

Demo 3

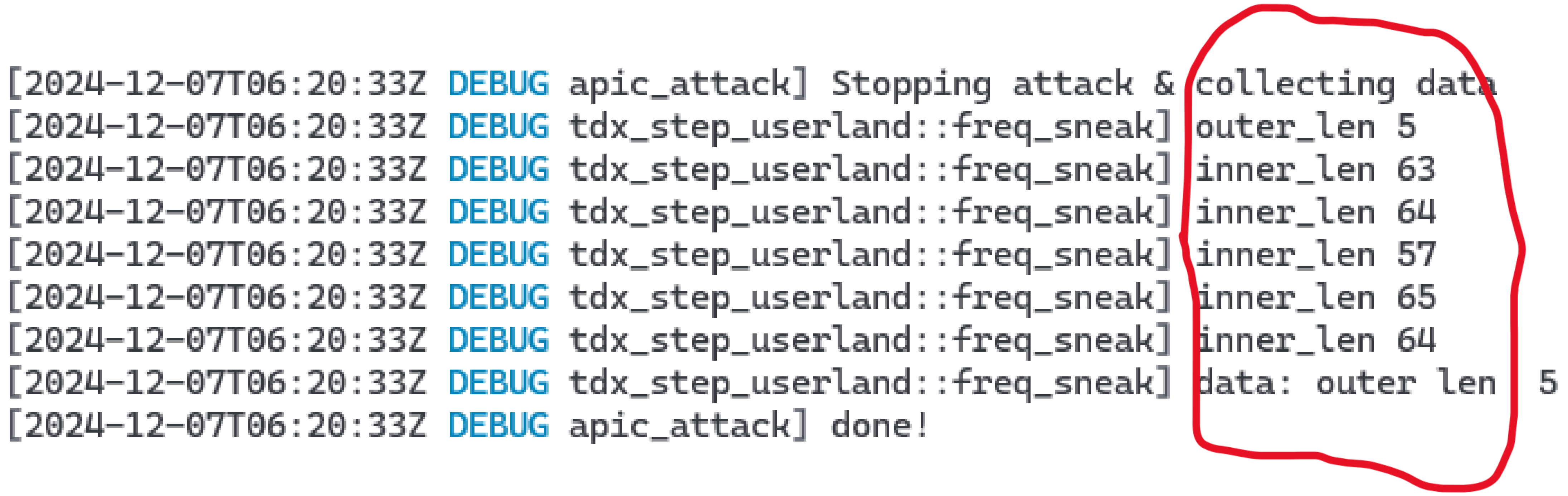
- Let’s attempt to single-step a TD by flushing the EPTs

```
// kmod-wbinvd.c
void apic_callback(void) {
     asm volatile ("\twbinvd\n" ::: "memory");
     fsleep(100000); // 100 ms
}
```

Listing 1: Evaluation target for single-stepping.

```
   mov   qword ptr[r8], 42
loop_label:
   dec rax
   nop
   nop
   nop
   nop
   jnz loop_label
   mov qword ptr [r9], 42
```

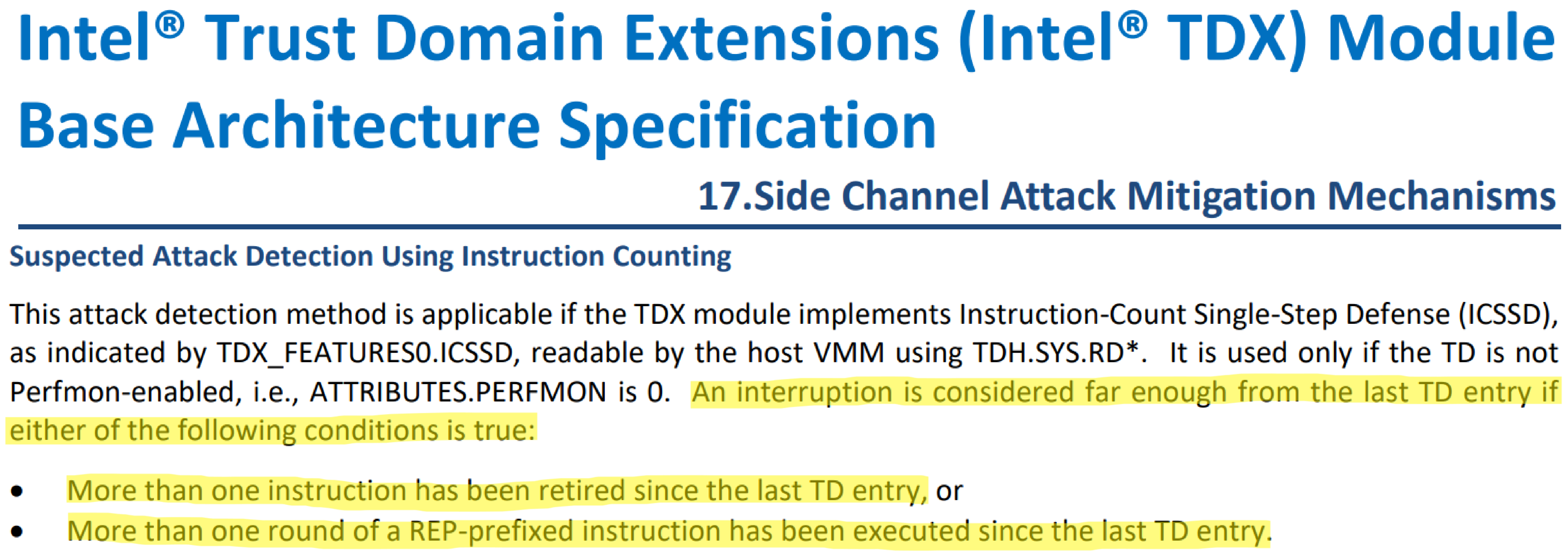
62 instructions



```
[2024-12-07T06:20:33Z DEBUG apic_attack] Stopping attack & collecting data
[2024-12-07T06:20:33Z DEBUG tdx_step_userland::freq_sneak] outer_len 5
[2024-12-07T06:20:33Z DEBUG tdx_step_userland::freq_sneak] inner_len 63
[2024-12-07T06:20:33Z DEBUG tdx_step_userland::freq_sneak] inner_len 64
[2024-12-07T06:20:33Z DEBUG tdx_step_userland::freq_sneak] inner_len 57
[2024-12-07T06:20:33Z DEBUG tdx_step_userland::freq_sneak] inner_len 65
[2024-12-07T06:20:33Z DEBUG tdx_step_userland::freq_sneak] inner_len 64
[2024-12-07T06:20:33Z DEBUG tdx_step_userland::freq_sneak] data: outer len  5
[2024-12-07T06:20:33Z DEBUG apic_attack] done!
```

Listing Source: L. Wilke, F. Sieck, and T. Eisenbarth, ‘TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX’, in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14--18, 2024*, 2024.

Instruction-Count Single-Step Defense



Intel® Trust Domain Extensions (Intel® TDX) Module Base Architecture Specification

17.Side Channel Attack Mitigation Mechanisms

Suspected Attack Detection Using Instruction Counting

This attack detection method is applicable if the TDX module implements Instruction-Count Single-Step Defense (ICSSD), as indicated by TDX_FEATURES0.ICSSD, readable by the host VMM using TDH.SYS.RD*. It is used only if the TD is not Perfmon-enabled, i.e., ATTRIBUTES.PERFMON is 0. An interruption is considered far enough from the last TD entry if either of the following conditions is true:

- More than one instruction has been retired since the last TD entry, or
- More than one round of a REP-prefixed instruction has been executed since the last TD entry.

Demo 4

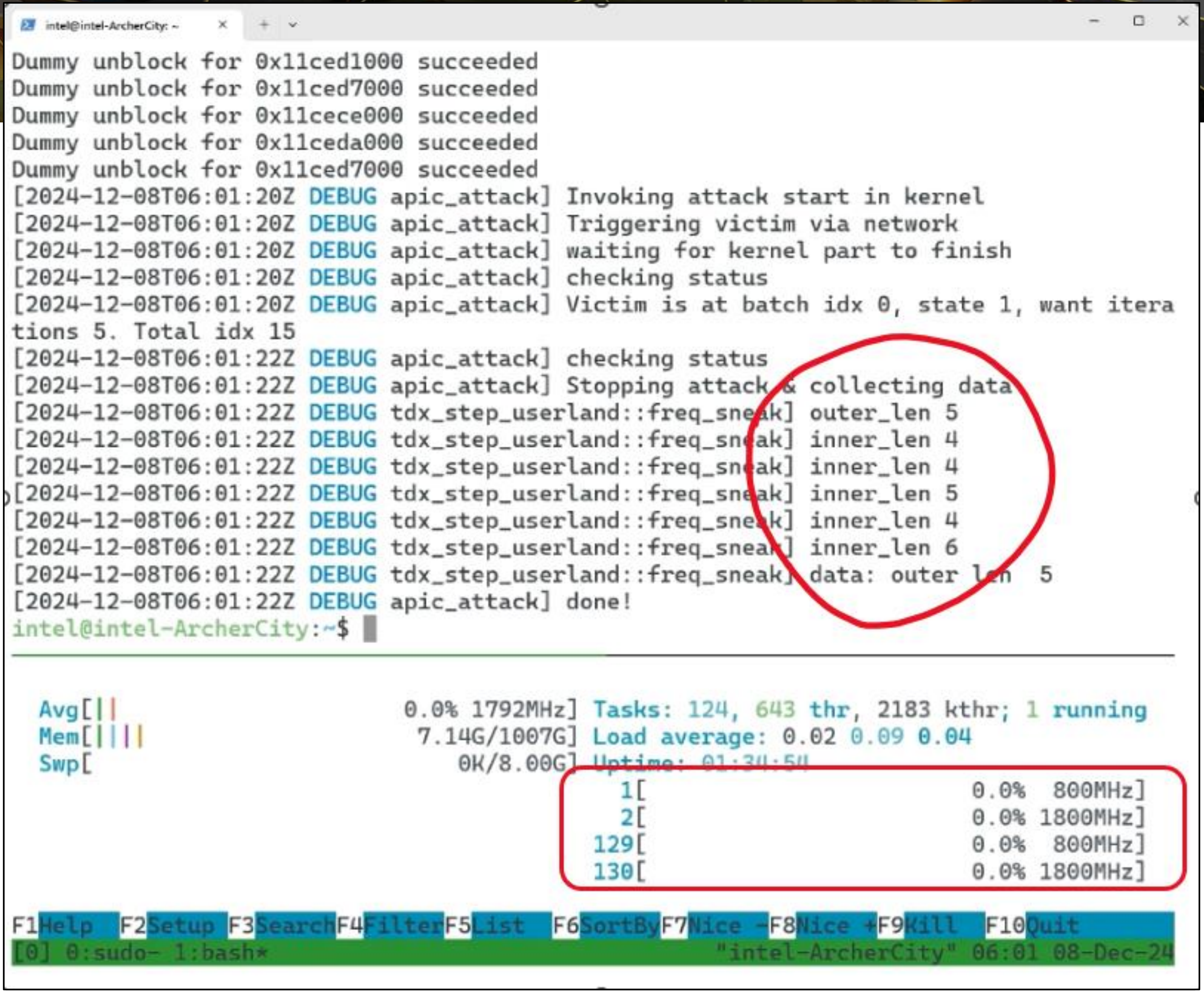
- Let's attempt to single-step a TD when ICSSD is enabled

Listing 1: Evaluation target for single-stepping.

```
   mov  qword ptr[r8], 42
loop_label:
   dec rax
   nop
   nop
   nop
   nop
   jnz loop_label
   mov qword ptr [r9], 42
```

62 instructions
÷
~16 instructions forced by the TDX module
=
~4 instructions per attack attempt

Listing Source: L. Wilke, F. Sieck, and T. Eisenbarth, 'TDXdown: Single-Stepping and Instruction Counting Attacks against Intel TDX', in *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14--18, 2024*, 2024.



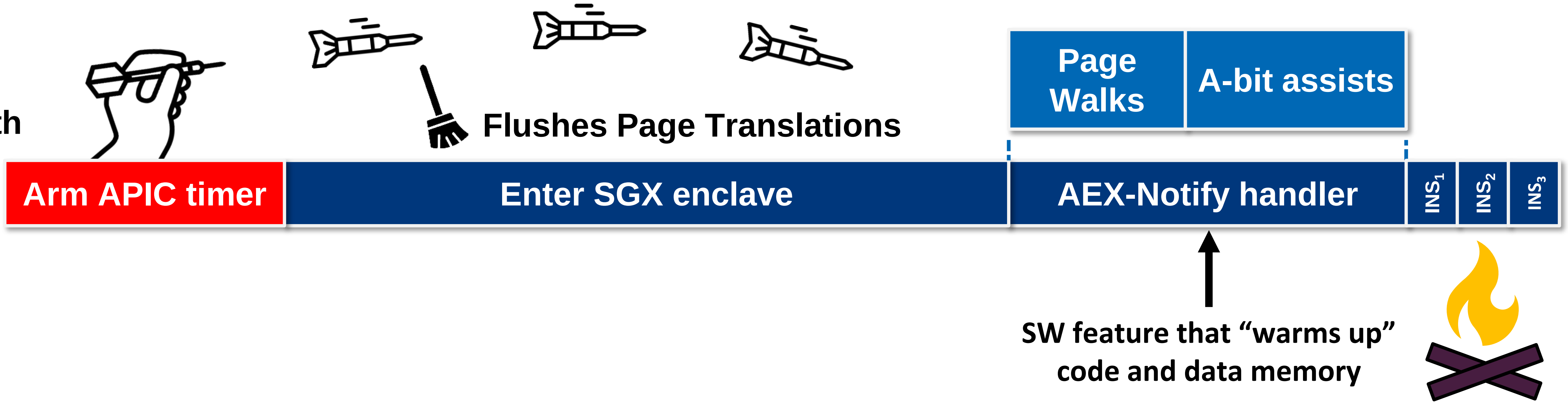
```
Dummy unblock for 0x11ced1000 succeeded
Dummy unblock for 0x11ced7000 succeeded
Dummy unblock for 0x11cece000 succeeded
Dummy unblock for 0x11ceda000 succeeded
Dummy unblock for 0x11ced7000 succeeded
[2024-12-08T06:01:20Z DEBUG apic_attack] Invoking attack start in kernel
[2024-12-08T06:01:20Z DEBUG apic_attack] Triggering victim via network
[2024-12-08T06:01:20Z DEBUG apic_attack] waiting for kernel part to finish
[2024-12-08T06:01:20Z DEBUG apic_attack] checking status
[2024-12-08T06:01:20Z DEBUG apic_attack] Victim is at batch idx 0, state 1, want itera
tions 5. Total idx 15
[2024-12-08T06:01:22Z DEBUG apic_attack] checking status
[2024-12-08T06:01:22Z DEBUG apic_attack] Stopping attack & collecting data
[2024-12-08T06:01:22Z DEBUG tdx_step_userland::freq_sneak] outer_len 5
[2024-12-08T06:01:22Z DEBUG tdx_step_userland::freq_sneak] inner_len 4
[2024-12-08T06:01:22Z DEBUG tdx_step_userland::freq_sneak] inner_len 4
[2024-12-08T06:01:22Z DEBUG tdx_step_userland::freq_sneak] inner_len 5
[2024-12-08T06:01:22Z DEBUG tdx_step_userland::freq_sneak] inner_len 4
[2024-12-08T06:01:22Z DEBUG tdx_step_userland::freq_sneak] inner_len 6
[2024-12-08T06:01:22Z DEBUG tdx_step_userland::freq_sneak] data: outer len  5
[2024-12-08T06:01:22Z DEBUG apic_attack] done!
intel@intel-ArcherCity:~$ 

  Avg[||                          0.0% 1792MHz]  Tasks: 124, 643 thr, 2183 kthr; 1 running
  Mem[||||                  7.14G/1007G]  Load average: 0.02 0.09 0.04
  Swp[                          0K/8.00G]  Uptime: 01:34:54
                                      1[                         0.0%  800MHz]
                                      2[                         0.0% 1800MHz]
                                    129[                         0.0%  800MHz]
                                    130[                         0.0% 1800MHz]

F1Help  F2Setup F3SearchF4FilterF5List  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
[0] 0:sudo- 1:bash*                             "intel-ArcherCity" 06:01 08-Dec-24
```

Agenda

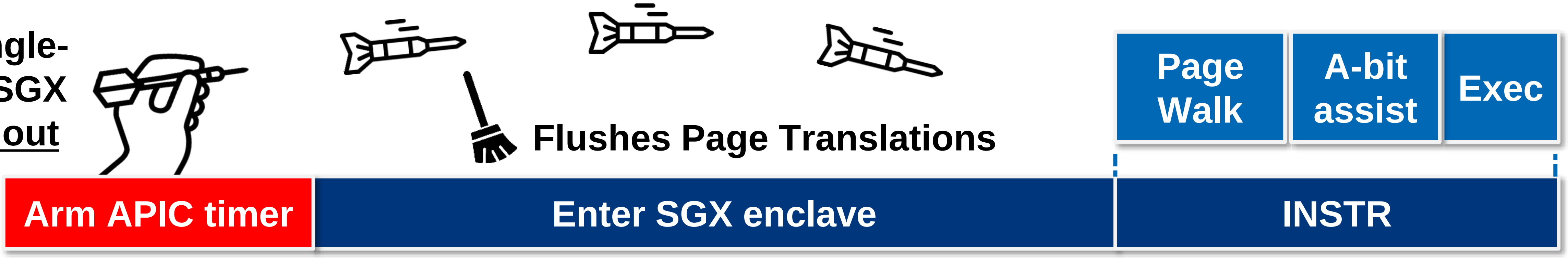
- Intro to Confidential Computing, Intel TDX (Trust Domain Extensions), side-channel Attacks, and malicious single-stepping
- Pre-TDX PoC (Proof of Concept) TDX-step exploit and mitigation
- Techniques to bypass the TDX-Step mitigation, and intro to the new ICSSD (Instruction Counting Single-Step Defense) feature
- Comparison with the SGX-Step mitigation

AEX-Notify: the SGX-Step mitigation

Example: An SGX enclave hardened with AEX-Notify

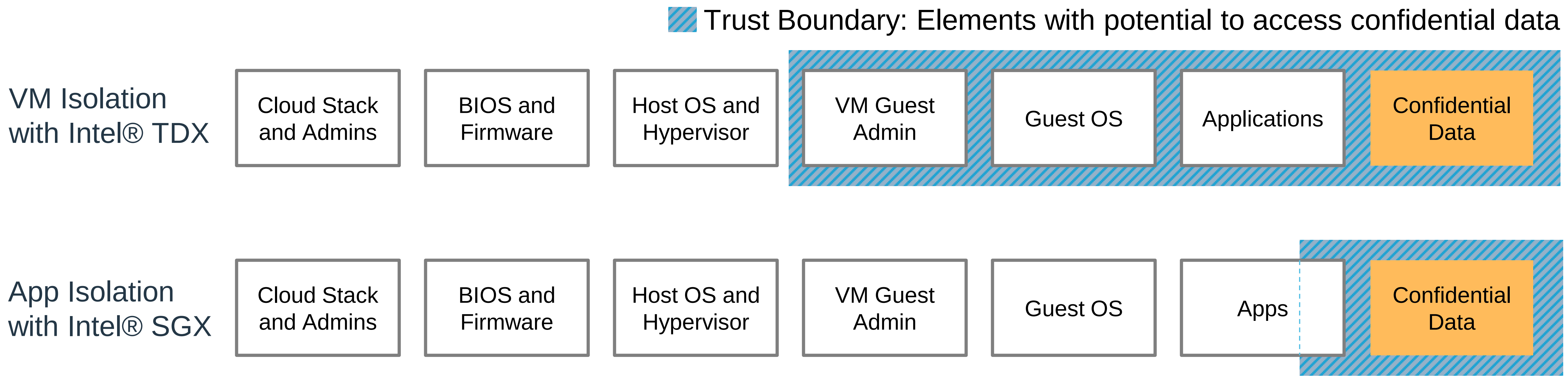


Example: Single-stepping an SGX enclave, <u>without</u> AEX-Notify



Why implement two different single-step mitigations for SGX and TDX?

- TDX is a virtualization-based technology, and therefore has different capabilities
- AEX-Notify uses a SW component that is available in the Intel SGX SDK—a similar SW solution for TDX would require para-virtualization in the guest OS
- Popular OS kernels such as Linux cannot handle arbitrary externally generated events



Conclusion (and Q&A before lunch)

- Confidential Computing's strong adversary model continues to drive exciting research. This presentation showed some capabilities that a privileged adversary may use:
  - Arming the APIC timer to hit an adversary-desired instruction with an interrupt
  - Scaling the clock frequency of the victim's CPU core
  - Flushing all of the CPU's caches
- Defense-in-depth mechanisms such as ICSSD and AEX-Notify can help to mitigate potential side-channel attacks against Trusted Execution Environments
- Modern processors use a variety of address translation caching techniques to dramatically accelerate memory accesses in virtualized environments

black hat
EUROPE 2024