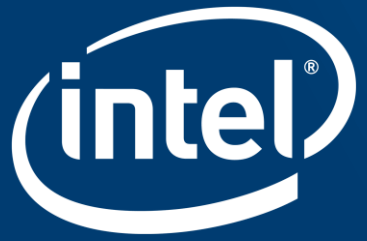


# HYPERVISOR-MANAGED LINEAR ADDRESS TRANSLATION

Gao Chao <chao.gao@intel.com>

October 30, 2020

# Agenda

Part1: Problem Statement

Part2: HLAT Introduction

Part3: Use HLAT to Enforce Guest Translation Integrity

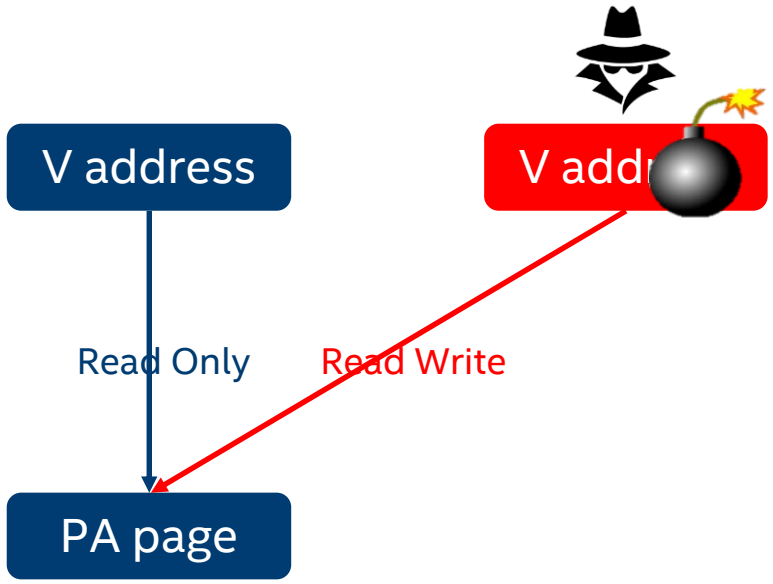
# PART1: PROBLEM STATEMENT

# Problem in Page Table Based Access Control

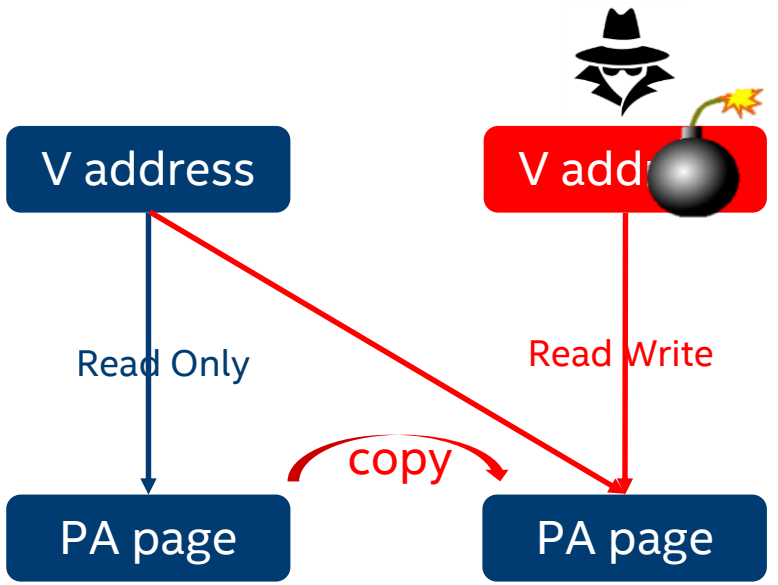
- Reduce attack surface through access control
  - E.g. executable code must not be writable
- Attackers can bypass access control by overriding page table
- Enforce address translation integrity
  - In a VM, write-protecting CR3 page table leads to high performance penalty

# Typical Page Table Overriding Attacks

## Alias Mapping




## Page Remapping



# PART2: HLAT INTRODUCTION

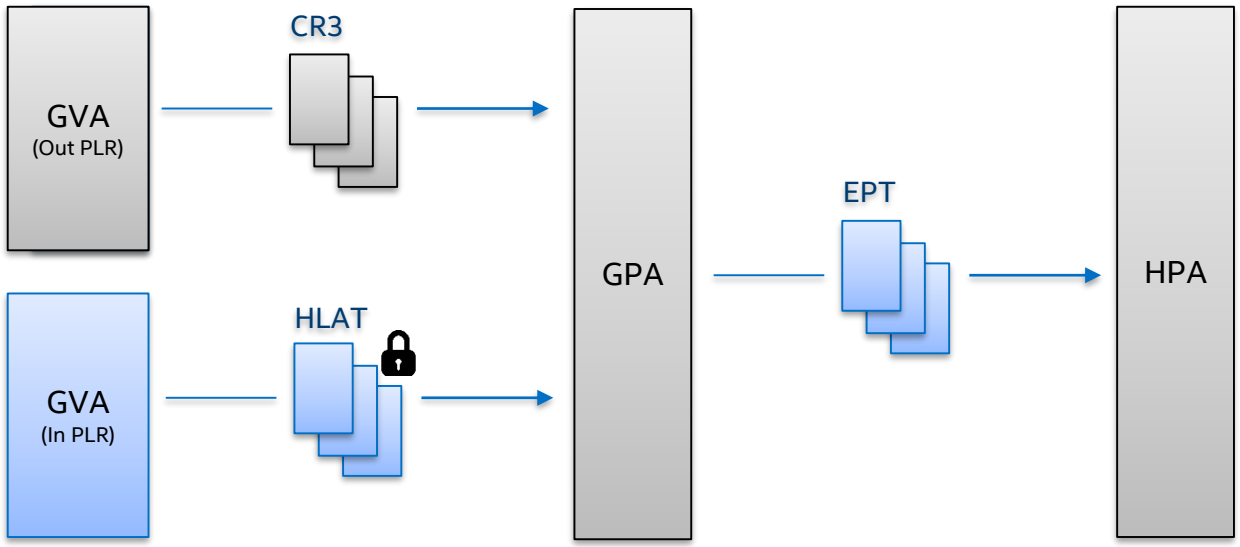
# HLAT Overview

SPEC: https://software.intel.com/content/www/us/en/develop/download/intel-architecture-instruction-set-extensions-programming-reference.html

Goal: enforce (guest) translation integrity with VMM

Protected linear range (PLR): a range of guest linear address space

**Key Idea**

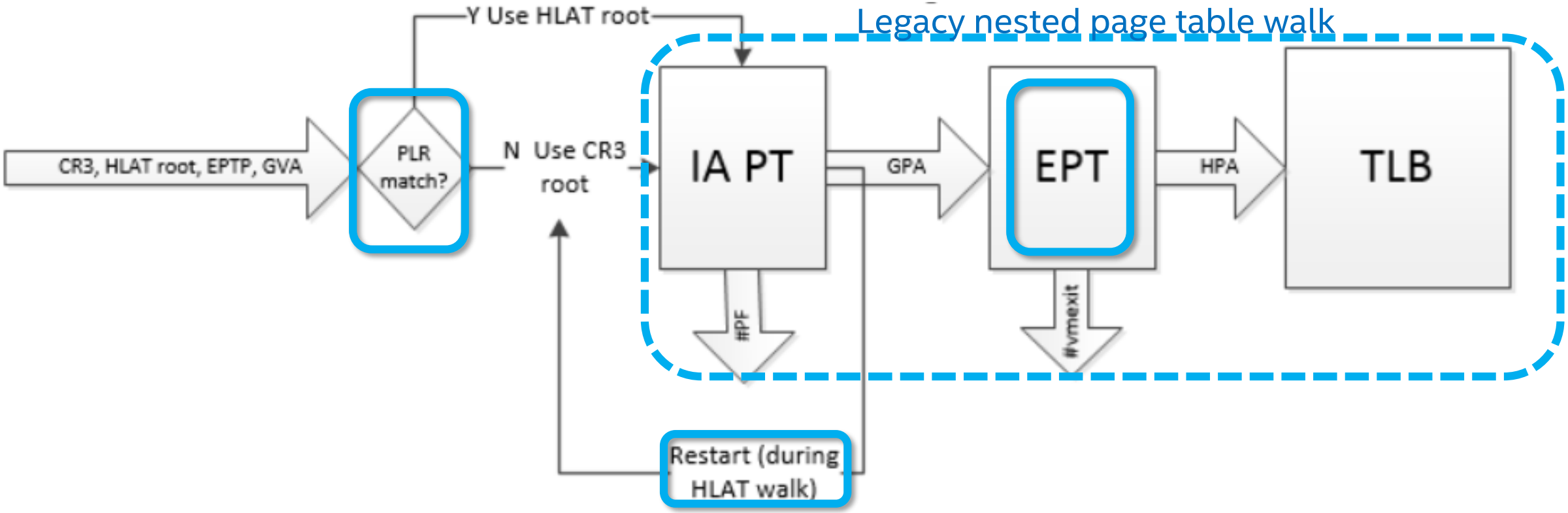



**Benefits**

Security: invulnerable to page remapping attack

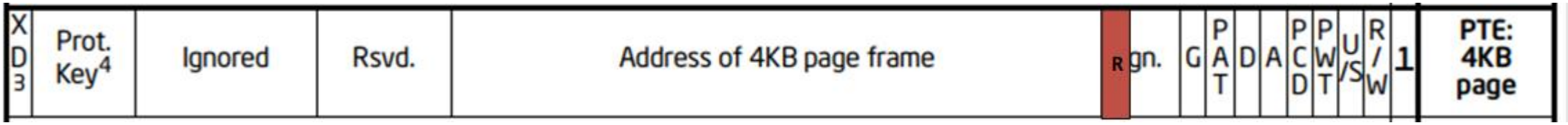
Efficiency: no interception to CR3 page table modification (compared with EPT-based page table protection).

# HLAT – Nested Page Table Walk



# HLAT – Paging Structures

- Same as IA32e paging structures
- Support both 5-level and 4-level paging
- Bit 11: “Restart”




## HLAT Terminal Fault

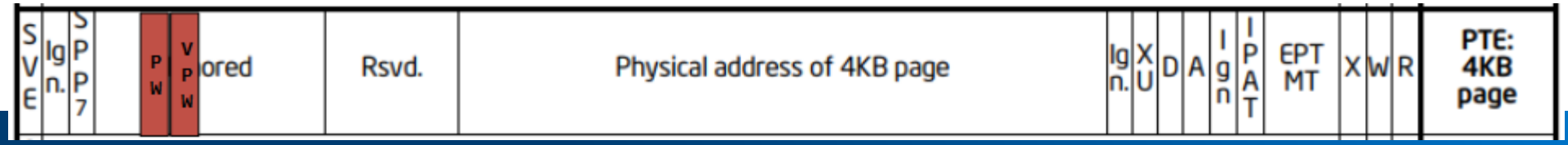
- non-present or reserved bits
- new page fault error code: bit 7 – “HLAT Terminal Fault”

# EPT Control Bit “Paging-Write”

- **“Paging Write”** allows CPU to update A/D bits on pages that are non-writable to software.

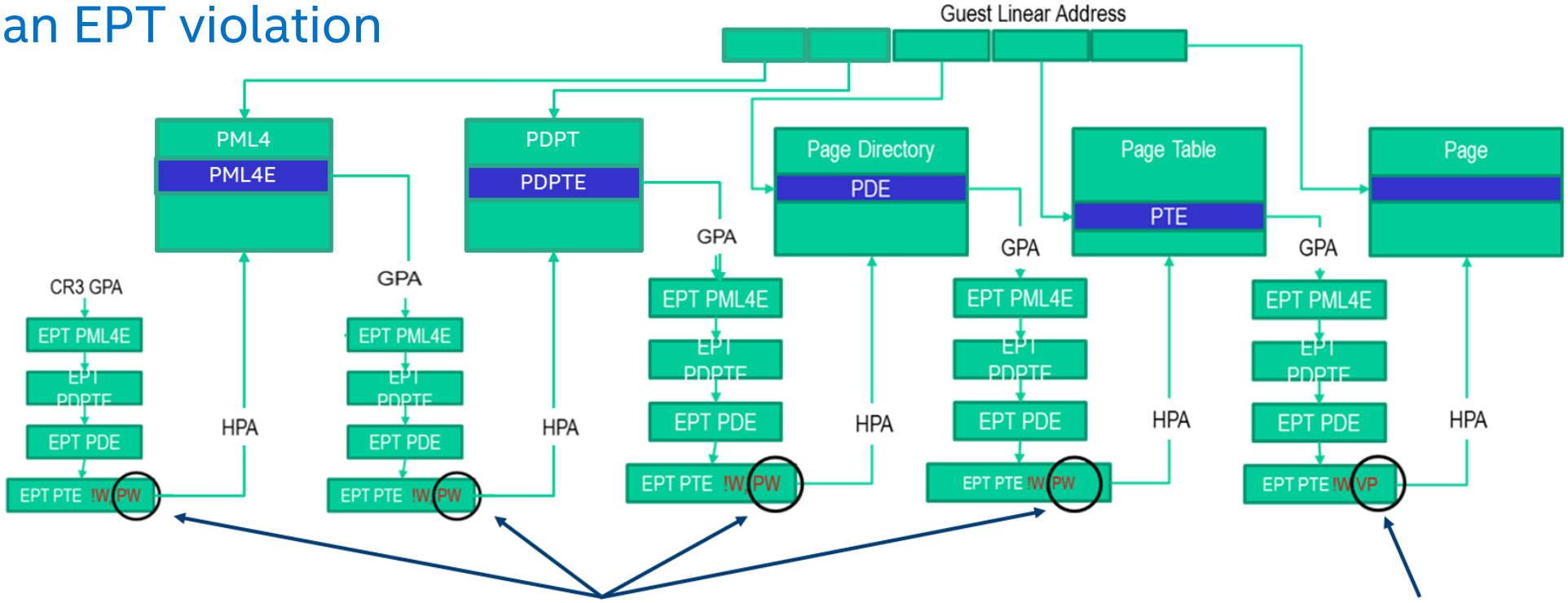
| EPT Control Bits | w | !w | !w\|pw |
|---|---|---|---|
| Software writes | Allowed | Denied | Denied |
| A/D bits update | Allowed | Denied | Allowed |

- Improve efficiency of read-only guest page table (under EPT):
  - Reduce VM-exits due to A/D bits update
  - Relieve VMM from A/D bits emulation



# EPT Control Bit “Verify Paging-write”

- “**Verify Paging-write”** enforces that all leaf guest-paging-structure pages encountered during the nested walk have PW set (under EPT), else generates an EPT violation

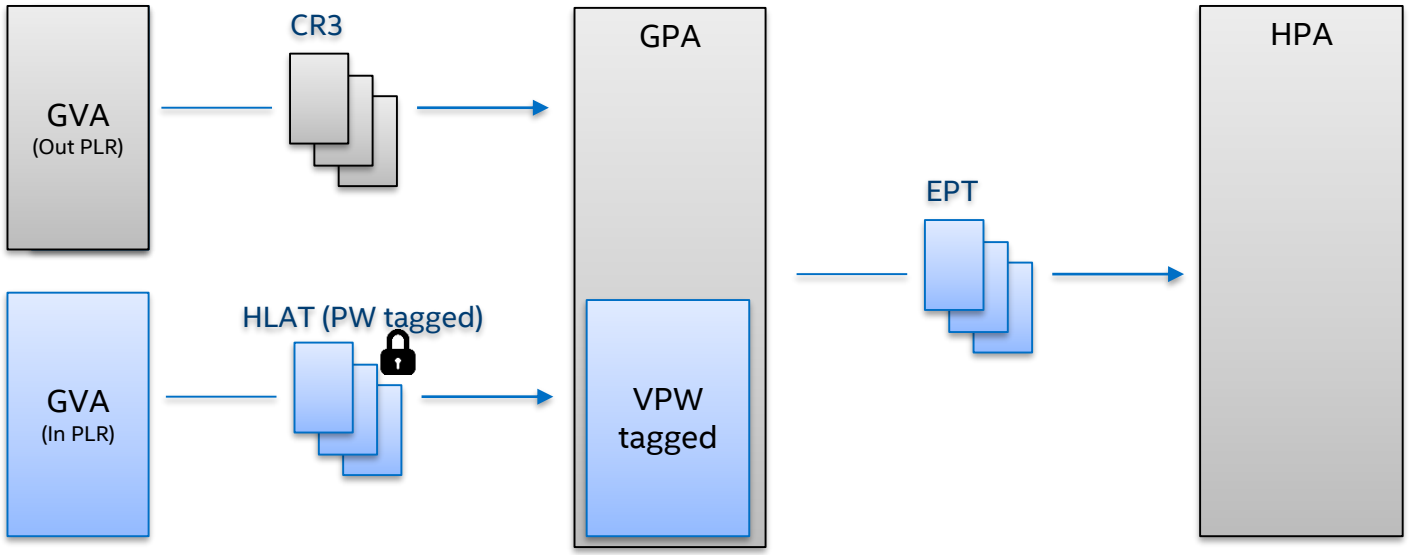



Enables VMM to complete CPU A/D bit updates on access-controlled Guest paging structure pages without EPT violation VM exits

Verifies that page-walk occurred through VMM access-controlled pages only

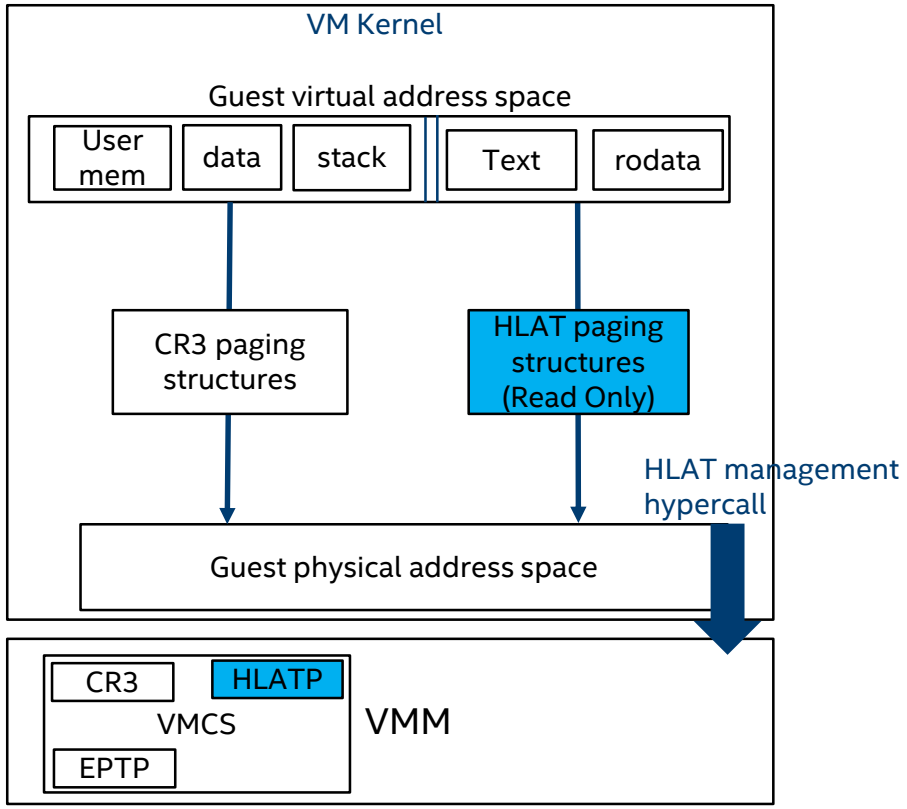
# Prevent Alias Mapping with PW & VPW

VPW tagged guest memory can only be accessed with HLAT (PW tagged page table)



# PART3: USE HLAT TO ENFORCE GUEST TRANSLATION INTEGRITY

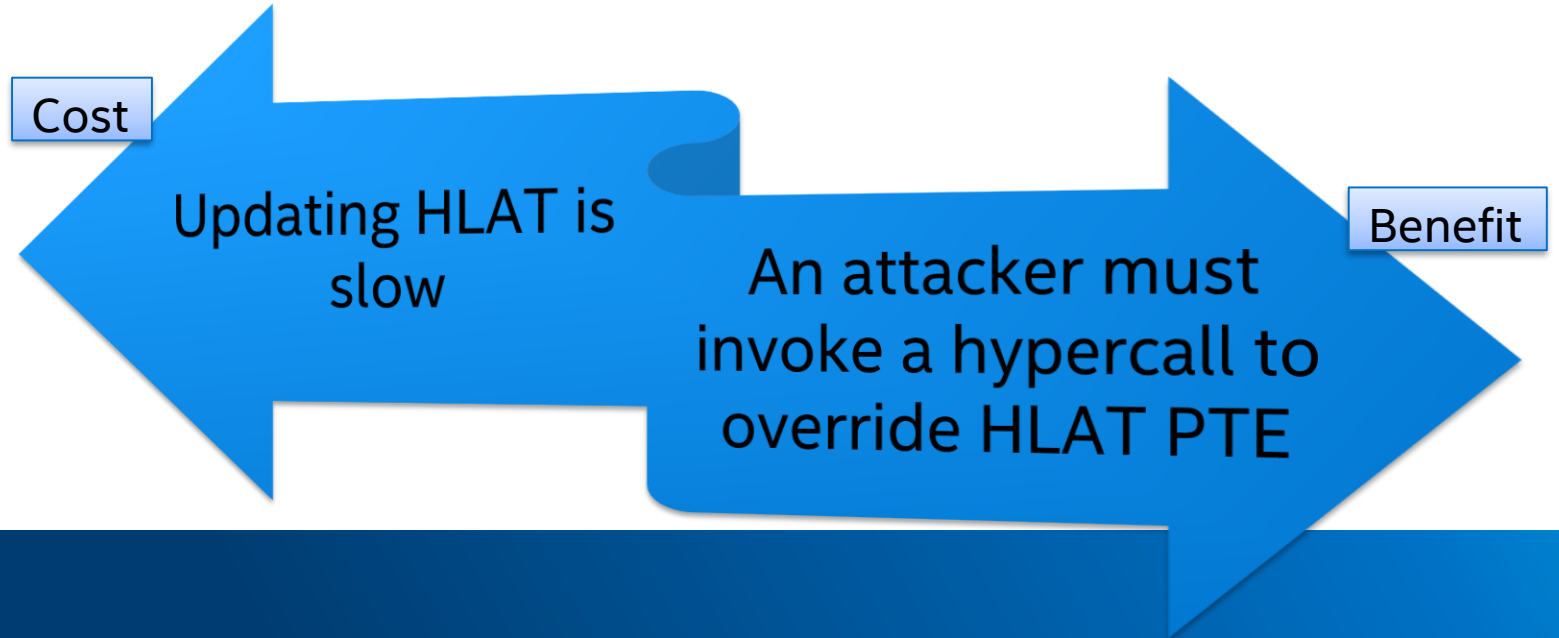
# Enforce Guest Translation Integrity




- Prevent page remapping
  - Write-protecting HLAT page table
- Prevent alias mapping
  - Set VPW for protected guest physical page under EPT
  - Set PW for HLAT paging structures pages under EPT

# Update HLAT page table

1. Revert write-protection on HLAT through hypercall
2. Update HLAT page table
3. enforce write-protection on HLAT through hypercall



# Security Value

In theory, an attacker with arbitrary memory write capability can make kernel text/rodata writable and then override them.

With this solution, an attacker cannot easily mark kernel text/rodata writable. It must first turn off HLAT or mark HLAT page table writable through hypercalls.

# Compare with Write-protecting CR3 Page Table

## Efficient

- Intercept CR3 switching
- Granularity (less impact to normal mappings)
  - 4KB vs. whole virtual address space

## Clean

- Small and less intrusive changes to memory management

# Demo

```c
static int __init kvmpat_test_init(void)
{
        int ret;

        printk("kvmpat test initing...\n");

        ret = set_memory_rw(target, 1);
        if (ret)
                printk("set memory to rw %d\n", ret);

        printk("attempt to write to %lx\n", target);
        *(unsigned long *)(void *)target = 0;

        return -1;
}
```

```
[   31.133307] kvmpat test initing...
[   31.133746] attempt to write to ffffffff95600000
[   31.133758] BUG: unable to handle page fault for address: ffffffff95600000
[   31.142005] #PF: supervisor write access in kernel mode
[   31.149670] #PF: error_code(0x0003) - permissions violation
[   31.156853] HLAT:
[   31.159396] PGD 1379c9023 P4D 1379c9023 PUD 1379ca023 PMD 1379cb023 PTE bbc00161
[   31.168156] PGD bd20e067 P4D bd20e067 PUD bd20f063 PMD 124aec063 PTE bbc00163
[   31.176611] Oops: 0003 [#1] SMP NOPTI
[   31.181160] CPU: 1 PID: 922 Comm: modprobe Not tainted 5.8.0+ #75
[   31.188437] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS 1.10.2-1ubun
[   31.199319] RIP: 0010:kvmpat_test_init+0x52/0x1000 [kvmpat_test]
[   31.206988] Code: 74 0e 89 c6 48 c7 c7 3c a0 3d c0 e8 eb 4f 32 d5 48 8b 35 41 d3 ff                                                                a0
f <48> c7 00 00 00 00 00 83 c8 ff 5d c3 00 00 00 00 00 00 00 00 00 00
[   31.228716] RSP: 0018:ffff9b7c807afc60 EFLAGS: 00010286
[   31.234946] RAX: ffffffff95600000 RBX: ffffffffc03de000 RCX: 0000000000000001
[   31.244416] RDX: 0000000000000000 RSI: ffff8961bbc98880 RDI: ffff8961bbc98888
[   31.254112] RBP: ffff9b7c807afc60 R08: 0000000000000001 R09: 00000000000001fd
[   31.264239] R10: ffff9b7c807afb40 R11: 00000000000001fd R12: 00000000ffffffff
[   31.274602] R13: ffff8961b6f80ef0 R14: 0000000000000000 R15: ffff9b7c807afe70
[   31.284659] FS:  00007f988baf3540(0000) GS:ffff8961bbc80000(0000) knlGS:0000000000000000
[   31.296642] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[   31.304795] CR2: ffffffff95600000 CR3: 0000000138ace003 CR4: 0000000000360ee0
[   31.314098] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[   31.324394] DR3: 0000000000000000 DR6: 00000000ffff07f0 DR7: 0000000000000400
[   31.334920] Call Trace:
[   31.338968]  do_one_initcall+0x52/0x210
[   31.344950]  ? _cond_resched+0x1a/0x50
[   31.351072]  ? kmem_cache_alloc_trace+0x1ad/0x230
[   31.357837]  ? __vunmap+0x1c3/0x220
[   31.363074]  do_init_module+0x5f/0x231
```

_text

Non-writable in HLAT. Writable in CR3

# Changes to KVM

## Advertise a PV feature

- CPUID hypervisor leaf

## New hypercalls

- Allow guest to set HLAT root page and PLR
- Allow guest to set VPW/PW/RO flags for guest pages
  - Extend page tracking mechanism

## Handle EPT violations due to guest's setting

- Report #VE to guest

# Changes to Guest Kernel

## Manage HLAT page table and EPT flags

- Hooks in set_memory_ro/rw APIs

## Handle #PF

- Handle HLAT terminal fault
- Walks HLAT

## Handle #VE

- In general, it means an attack is detected

# Status and Plan

## Status

- Finished changes on KVM/guest kernel and some tests in kvm-unit-tests
- Verified on simulator

## Plan

- Send out RFC patches
- Explore the possibility of using HLAT to enforce integrity for non-executable mapping

# THANKS! Q&A