

AMD Secure vIOMMU Architecture and Implementation Challenges

Vasant Hegde <vasant.hegde@amd.com>

Suravee Suthikulpanit <Suravee.Suthikulpanit@amd.com>

LINUX
PLUMBERS
CONFERENCE

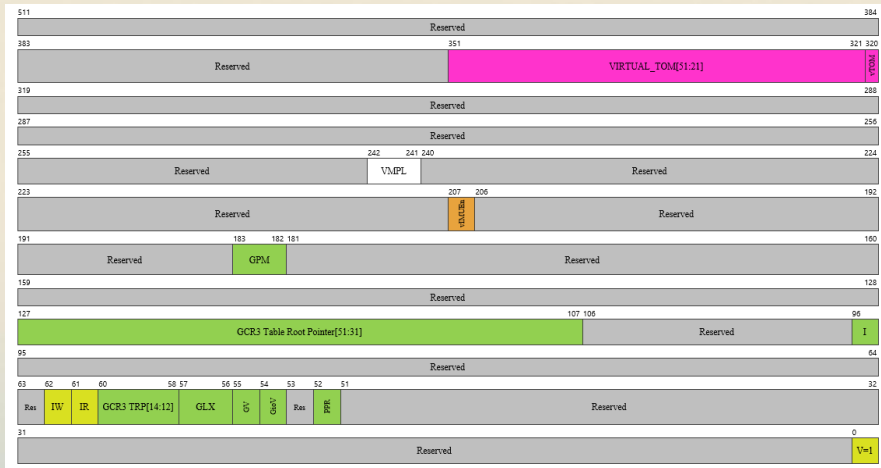
PRAGUE, CZECHIA
OCT. 5-7, 2026

Secure vIOMMU

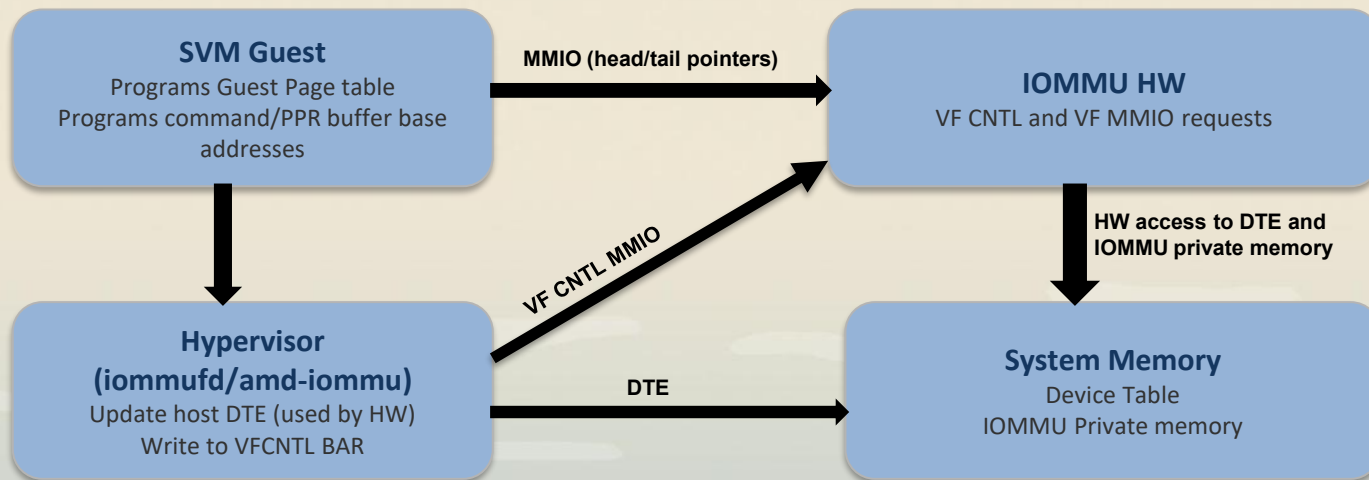
- Extends hw-vIOMMU to assign trusted IO (TIO) devices to confidential (SNP) guests
- Enables secure DMA directly between the trusted device and guest private memory
- Keeps the **untrusted hypervisor** off the secure path: AMD Security Processor (**ASP**) owns the secure DTE and guest IOMMU configuration
 - VF Control read/write from hypervisor is blocked for secure guests

Secure DTE

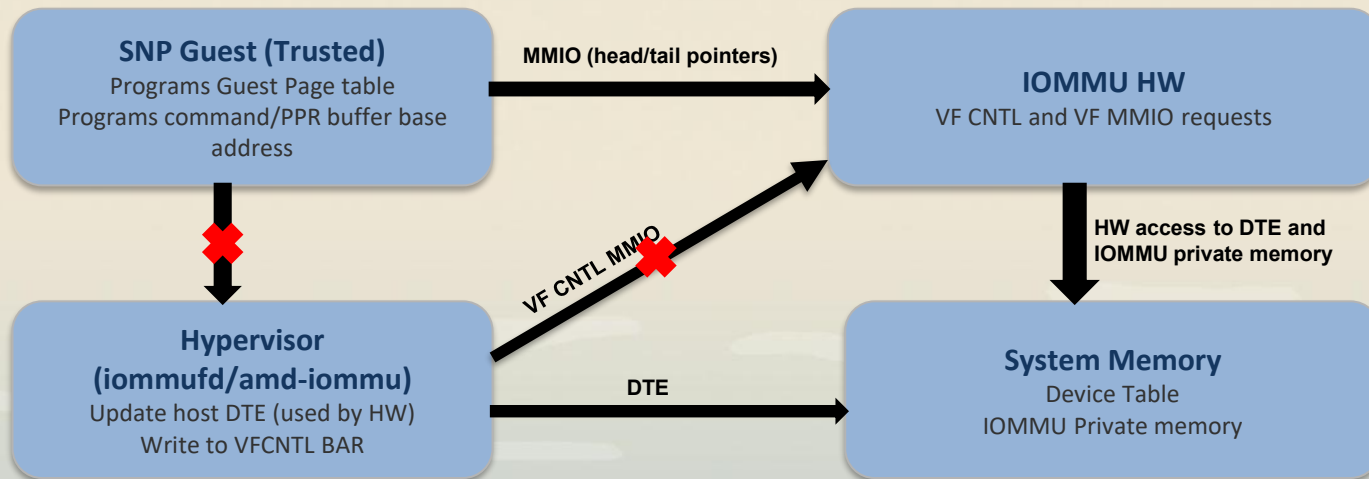
- Separate device table holding the DTEs of trusted devices
- 512-bit entry per device containing guest page table details, etc.
- Owned and programmed exclusively by the **ASP**, in encrypted memory
- Hypervisor cannot read/write to sDTE



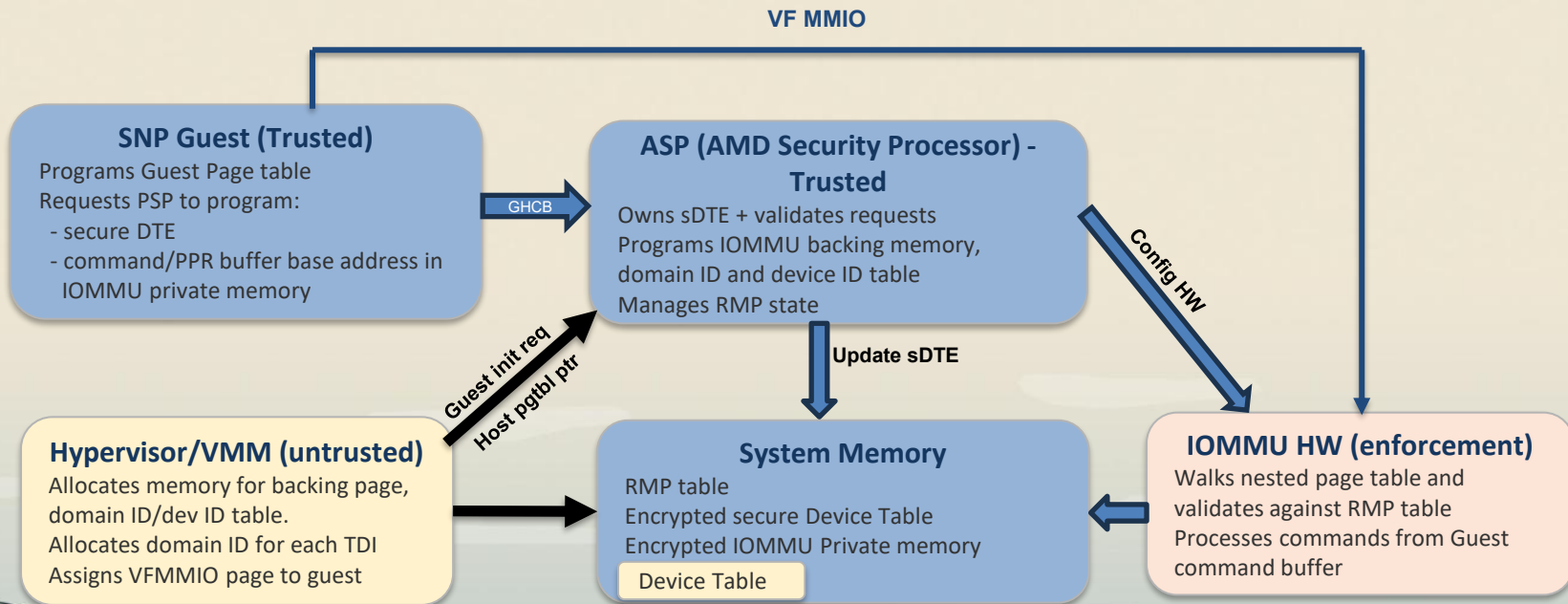
HW-vIOMMU setup



Secure vIOMMU setup



Secure vIOMMU setup



Current Status

- Working prototype of Secure vIOMMU is ready
 - **Kernel:** https://github.com/AMDESE/linux-iommu/tree/sviommu/tsm0407_v619_v1
 - **QEMU:** https://github.com/AMDESE/qemu-iommu/tree/sviommu/tsm0407_v619_v1
- Builds on the AMD hw-vIOMMU series (under upstream review)
- Plan to post to the upstream mailing list once hw-vIOMMU feature is accepted
- Early feedback on the design and interfaces is welcome

Challenge #1

- Both host and guest support resides in the **AMD IOMMU driver**, which uses ops structures to interact with other subsystems (sev_guest, crypto/ccp).

- Guest ops to communicate with the sev_guest subsystem:

```
struct amd_sviommu_guest_ops {  
    int (*setup_cmdbuf)(u16 devid, void *data);  
    int (*setup_mmio)(u16 devid, void *data);  
    int (*mapping_update)(void *data);  
    int (*sdte_update)(void *data);  
};
```

- Host ops to communicate with the ccp (crypto) subsystem:

```
struct amd_iommu_ccp_ops {  
    int (*sev_tio_viommu_init)(struct amd_sviommu *sv);  
    int (*sev_tio_viommu_guest_init)(struct amd_sviommu_guest *g);  
    int (*sev_tio_viommu_guest_shutdown)(struct amd_sviommu_guest *g);  
};
```

- An alternative is to use EXPORT functions.

Challenge #2

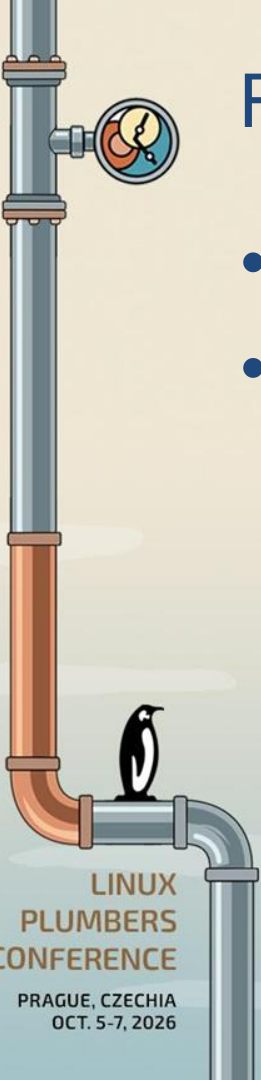
- Initialize secure vIOMMU instance
 - The hypervisor communicates with the ASP to initialize the secure vIOMMU instance.
 - A reference to **kvmfd** (struct kvm) is required in the VIOMMU_INIT path.
 - It is needed to obtain the SNP guest context page, ASID, etc. (struct kvm_sev_info)
 - The current prototype passes kvmfd via the IOMMU_VIOMMU_ALLOC uAPI
 - Plan to move to the new *viommu_init* interface, which provides an iommufd_device structure holding the *kvm_file* reference ([based on Aneesh's RFC v6](#)).
 - The ASP needs the **guest vIOMMU device ID** to map it to the physical IOMMU
 - The current prototype passes it via the IOMMU_VIOMMU_ALLOC uAPI.

Challenge #3

- **Synchronizing** the IOMMU host page table with the CPU page table
 - IOMMU hardware checks the RMP (Reverse Map Table) to verify page ownership before allowing DMA.
 - The IOMMU host page table entry **size** must match the corresponding RMP entry.
 - The IOMMU host page table must be synchronized before DMA from a secure guest is enabled.
 - The current prototype uses the generic_pt cut_mapping() interface to split the page table entry (Jason's prototype code).
 - Any other better way to handle this?

Reference

- [AMD IOMMU specification](#)
- [SEV TIO specification](#)



LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

COPYRIGHT AND DISCLAIMER

©2026 Advanced Micro Devices, Inc. All rights reserved.

AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

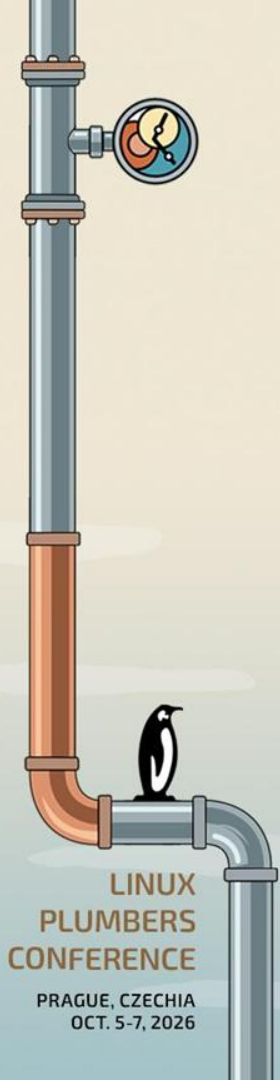
The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions, and typographical errors. The information contained herein is subject to change and may be rendered inaccurate releases, for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. Any computer system has risks of security vulnerabilities that cannot be completely prevented or mitigated. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

THIS INFORMATION IS PROVIDED 'AS IS.' AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS, OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION. AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY RELIANCE, DIRECT, INDIRECT, SPECIAL, OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Backup

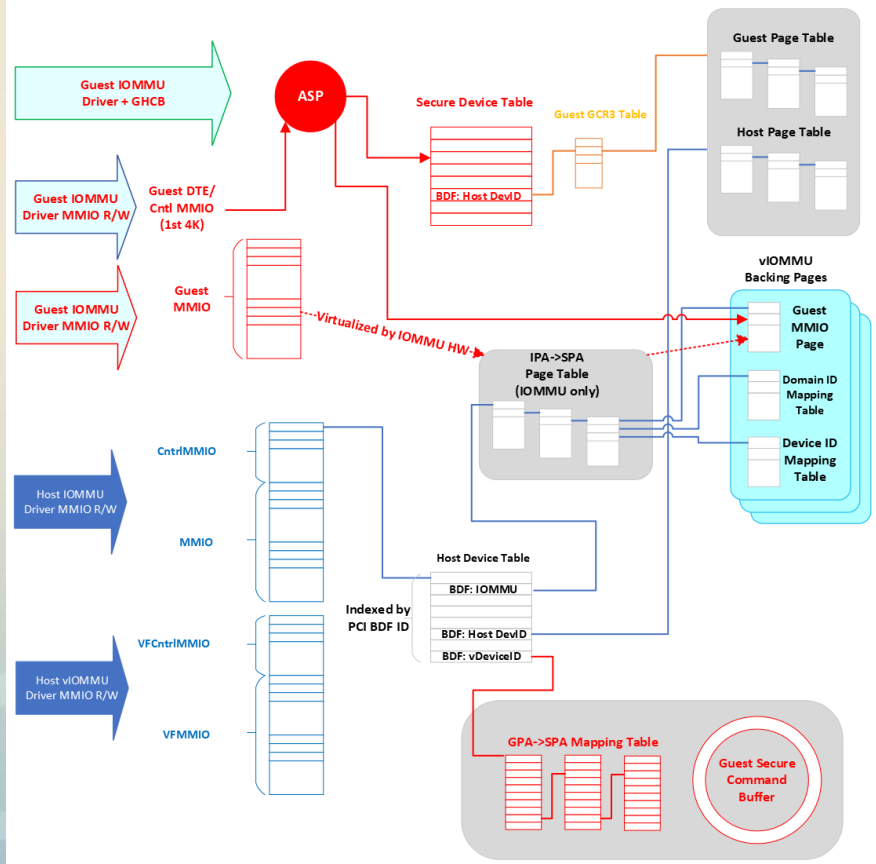


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

Secure vIOMMU Architecture

SNP Guest w/ TDISP Device

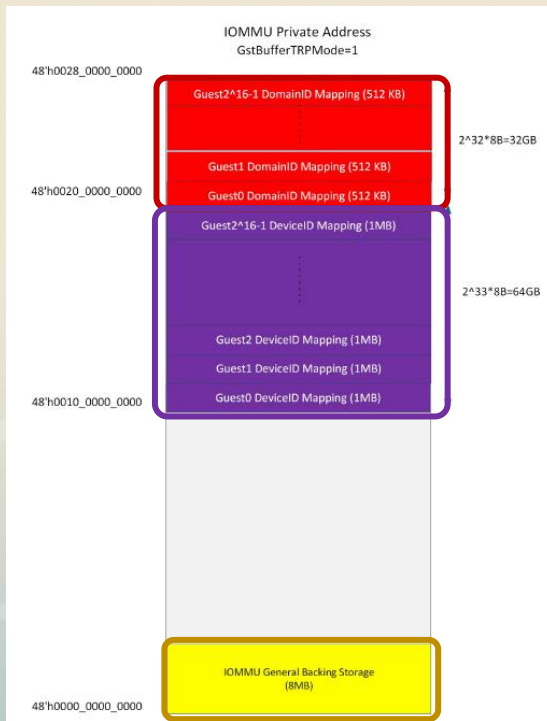


LINUX
PLUMBERS
CONFERENCE

PRAGUE, CZECHIA
OCT. 5-7, 2026

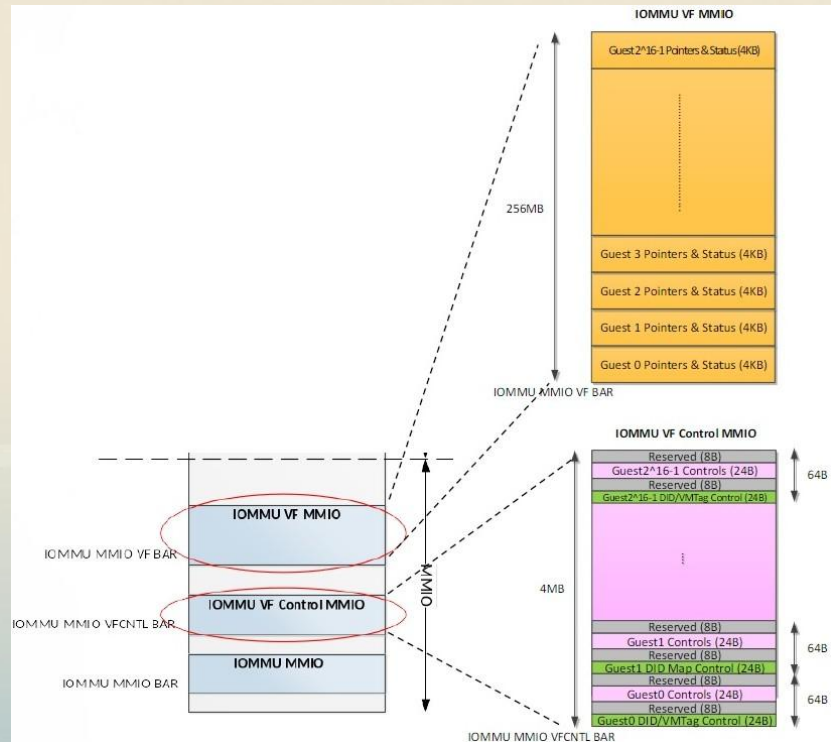
IOMMU Private Address Space

- Page table at IOMMU BDF used by IOMMU hardware to keep the vIOMMU config data
- Supports upto 64K guests
 - Uses Guest ID to index various tables
 - Top 32K IDs reserved for secure guests
- Per guest mapping table
 - **Guest Domain ID to Host domain ID mapping table**
 - **Guest Device ID to Host device ID mapping table**
- **Global 8MB per IOMMU for private data including per guest status, etc.**



VF Control and VF MMIO BAR

- **VF Control MMIO BAR**
 - Control bits, programmed by the hypervisor
 - 1st 4KB of the guest MMIO region
- **VF MMIO BAR**
 - 3rd 4KB of the guest MMIO region – virtualized by HW
 - Uses Guest ID to index/map 4K region to guest
 - Contains:
 - Command buffer head/tail pointers
 - Event and PPR log head/tail pointers
 - Status registers



Nested Page table

Host Device Table Entry

255	248	247	246	245	244	240	239	224
SnoopAttribute				Mode/FC	AttrV	Reserved	GuestID[15:0]	
223	GDeviceID[15:0]					virtEn	Reserved	
191	190	189	188	187	186	185	184	183
182	181	180	179	Interrupt Table Root Pointer [51:32]				
159	158	157	156	155	154	153	152	128
Interrupt Table Root Pointer [31:6]								IG IntTabLen IV
127	126	125	124	123	122	121	120	96
GCR3 Table Root Pointer [31:3]								SA SE I
95	94	93	92	91	90	89	88	64
GCR3 Table Root Pointer [30:1]								DomainID[15:0]
63	62	61	60	59	58	57	56	32
31	30	29	28	27	26	25	24	0
Host Page Table Root Pointer [61:32]								Reserved
31	30	29	28	27	26	25	24	0
Host Page Table Root Pointer [31:32]								TV V

GCR3 Table Root Pointer - vIOMMU handles guest translation

Host Page Table Root Pointer - IOMMU handles nested translation

Nested Page table

