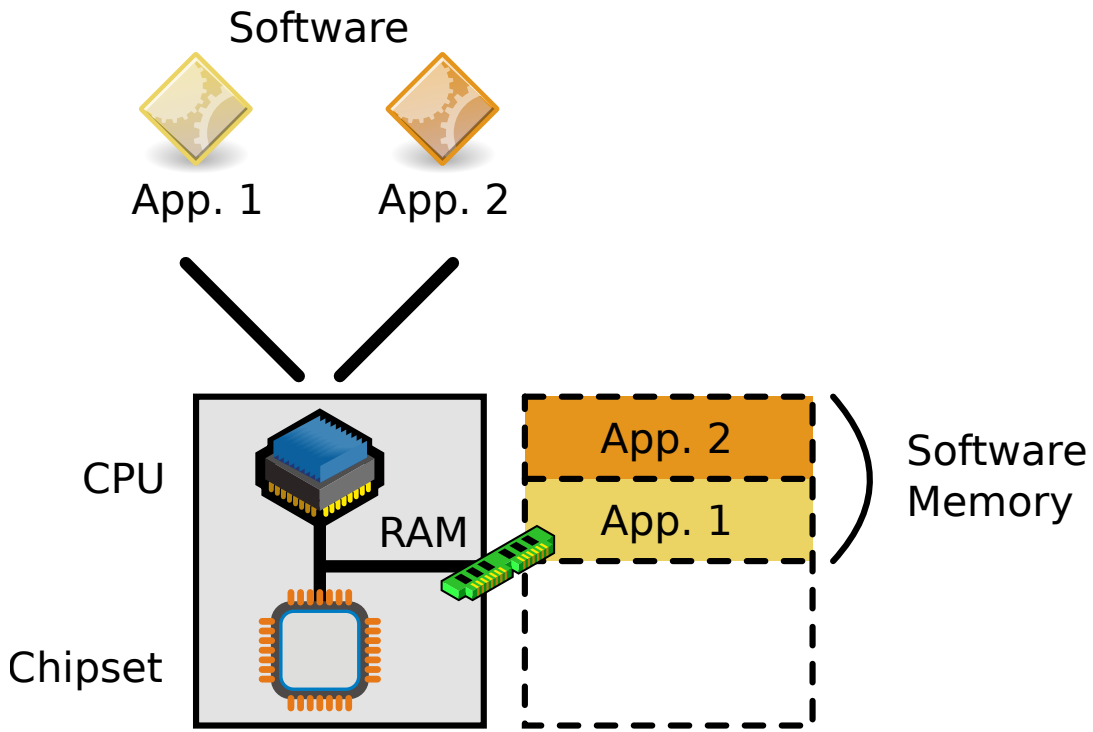
Bypassing IOMMU protection against I/O Attacks

Benoît Morgan, Éric Alata,
Vincent Nicomette and Mohamed Kaâniche

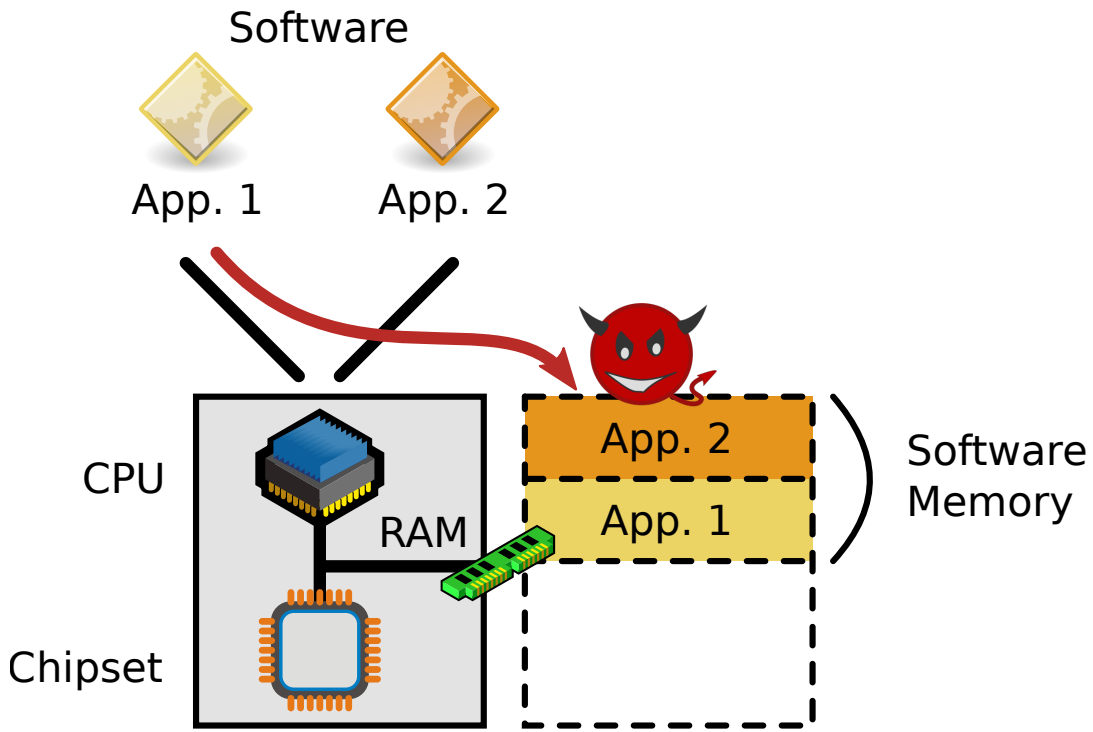
LAAS-CNRS, INSA Toulouse, University of Toulouse

October 19, 2016

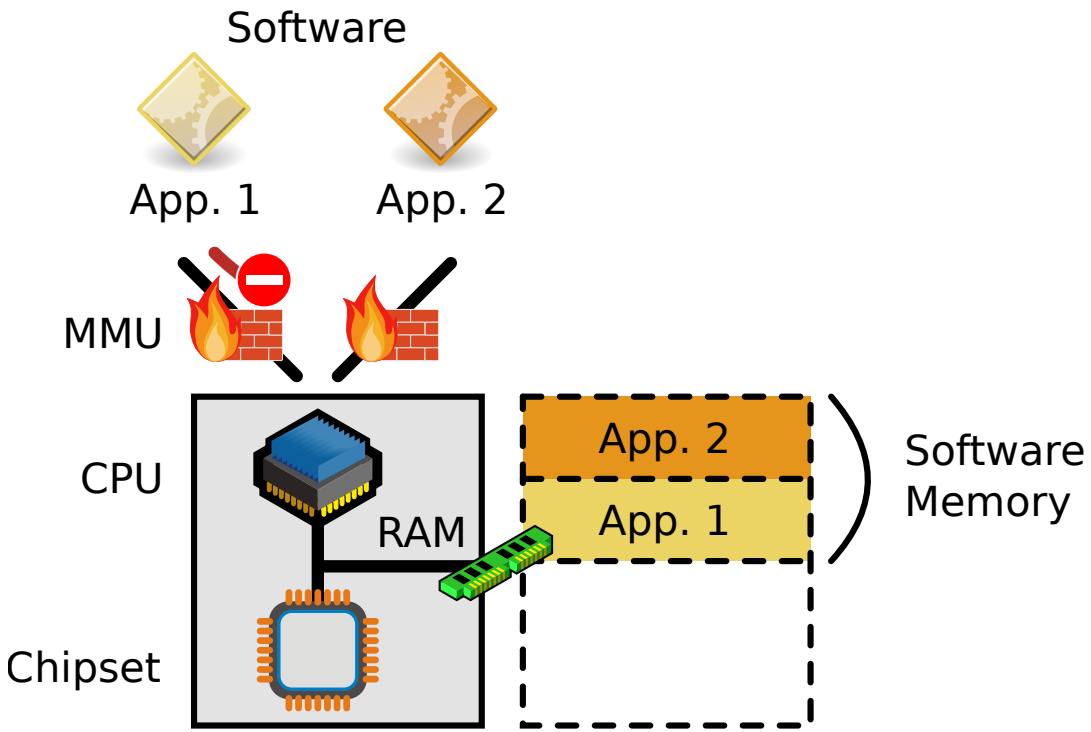
Memory protection in modern architectures



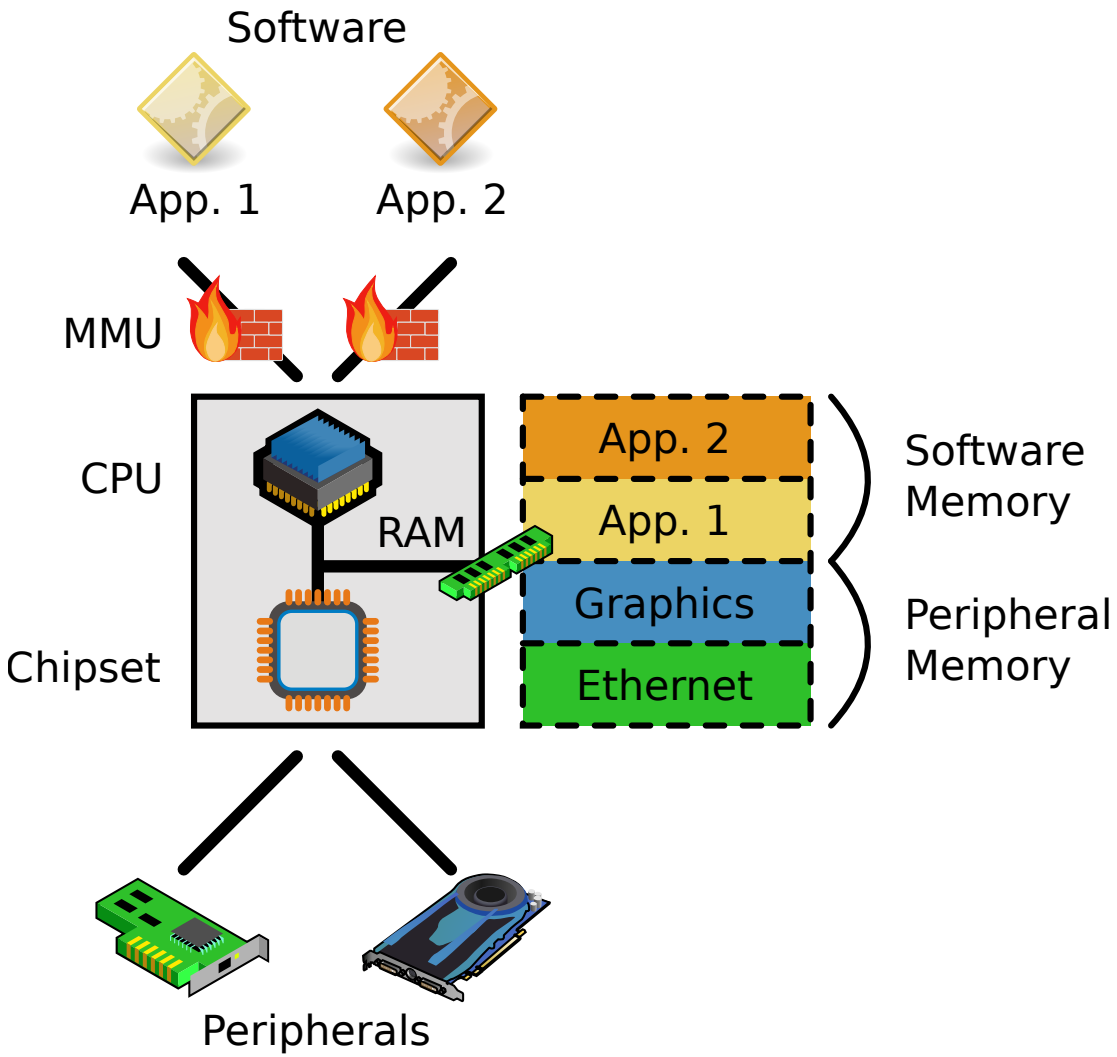
Memory protection in modern architectures



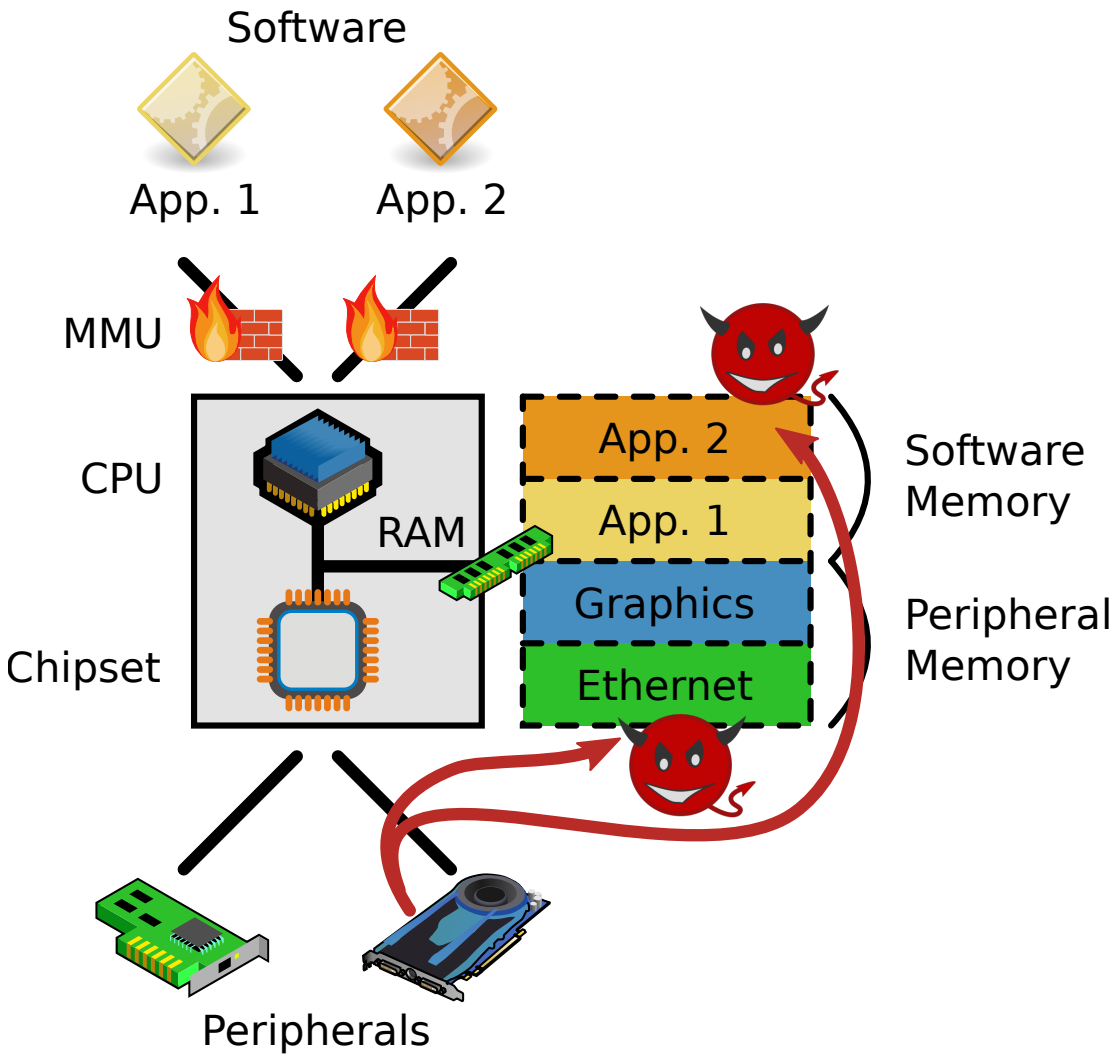
Memory protection in modern architectures



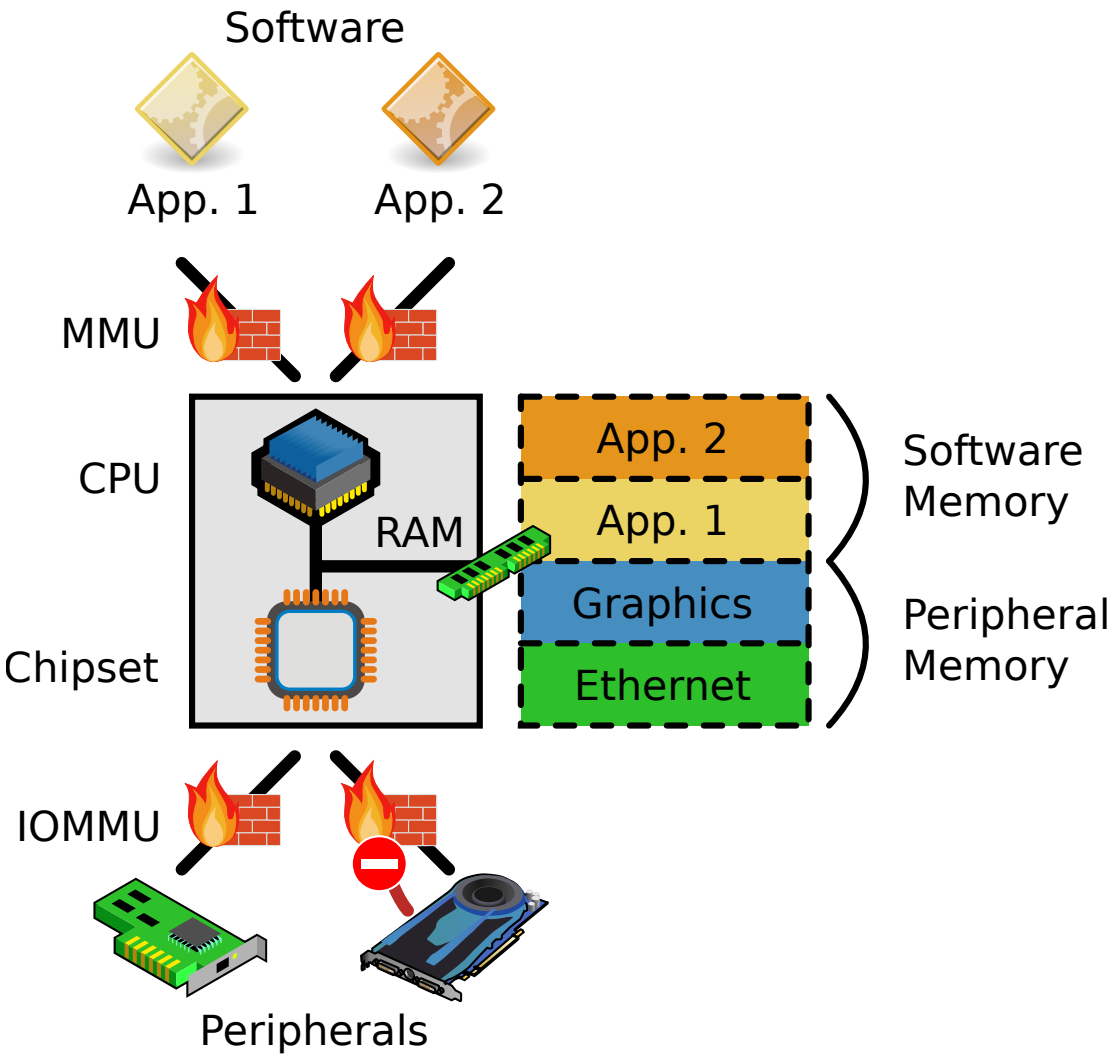
Memory protection in modern architectures



Memory protection in modern architectures



Memory protection in modern architectures



IOMMU security issues

- Do modern operating systems use this component by default ?
- Is it used correctly ?
- Is it active all the time ?

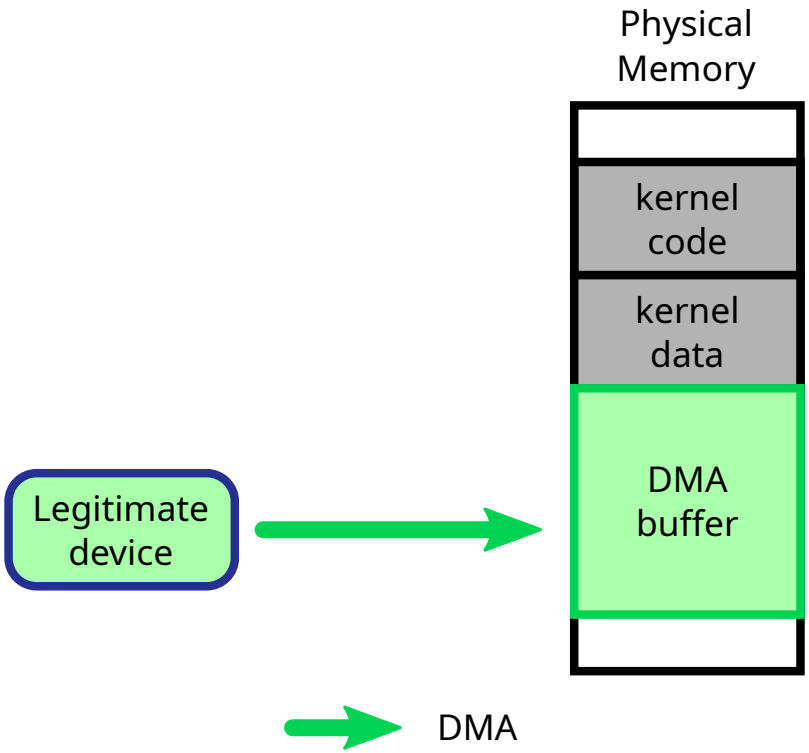
Contribution

- Vulnerability in the IOMMU configuration and exploitation proof of concept

Outline

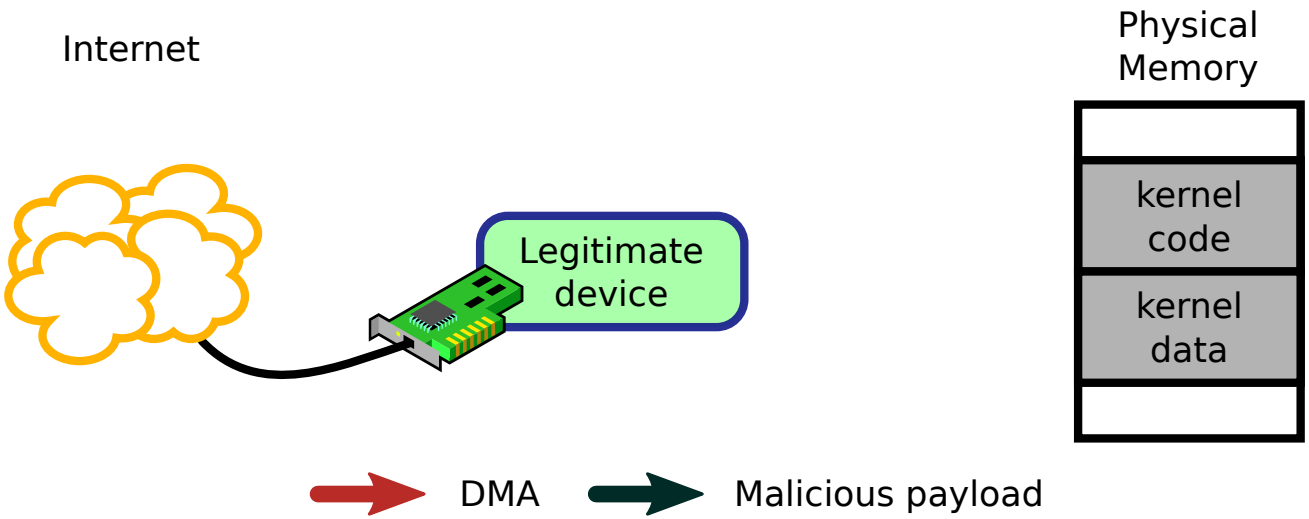
1. Background
   - DMA principles
   - IOMMU principles
2. A vulnerability in the IOMMU configuration
3. Conclusion

Direct Memory Access



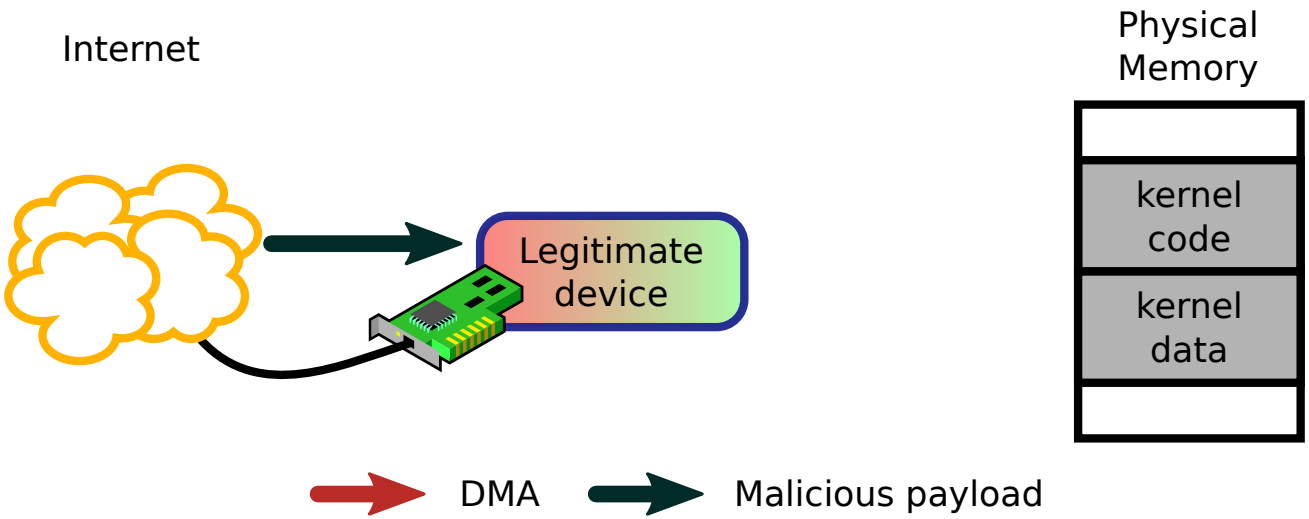
- Peripheral access to the main memory
- Independent from the CPU

DMA threats



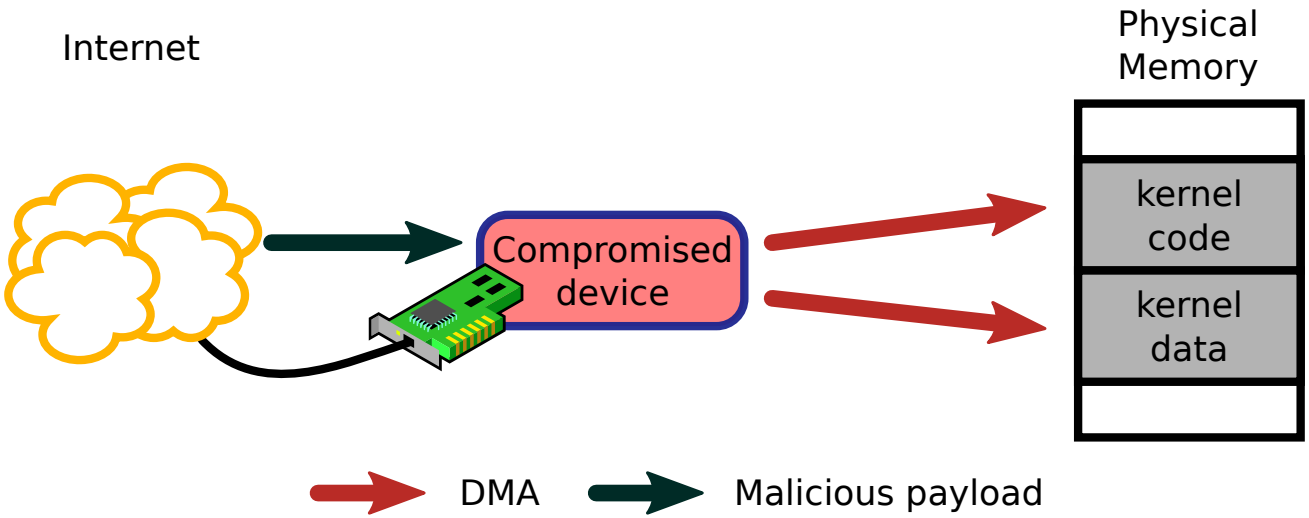
- Some peripherals are vulnerable and remotely exploitable (*Duflot et al.* [1])

DMA threats



- Some peripherals are vulnerable and remotely exploitable (*Duflot et al.* [1])

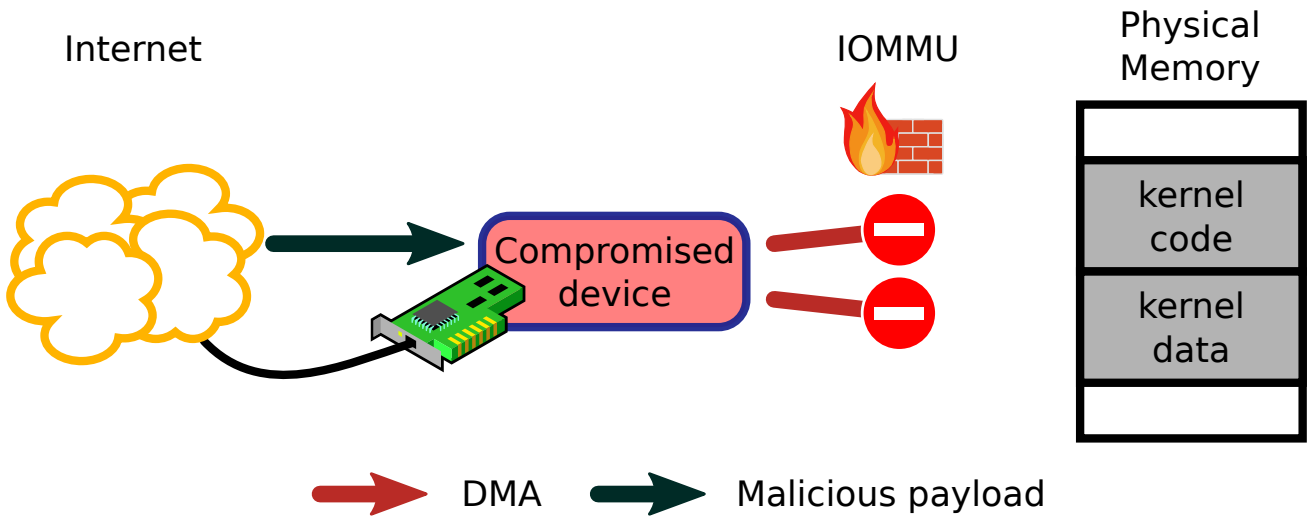
DMA threats



- Some peripherals are vulnerable and remotely exploitable (*Duflot et al.* [1])

$\Rightarrow$ Remote malicious arbitrary reads and writes of privileged data

DMA threats



- Some peripherals are vulnerable and remotely exploitable (*Duflot et al.* [1])

$\Rightarrow$ Remote malicious arbitrary reads and writes of privileged data

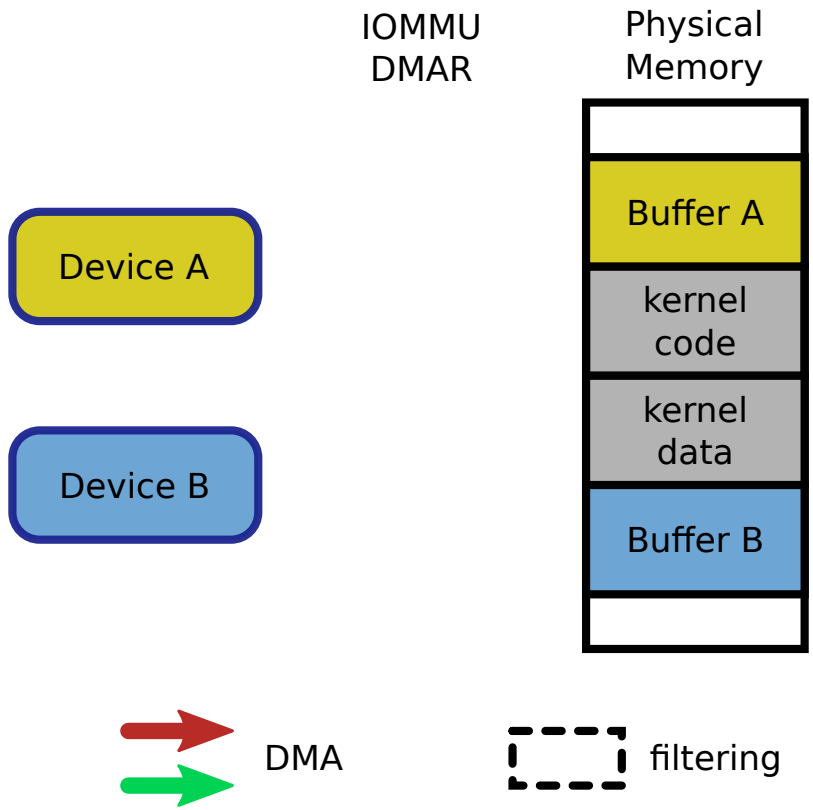
$\Rightarrow$ Access control is essential

Input Output Memory Management Unit

Services

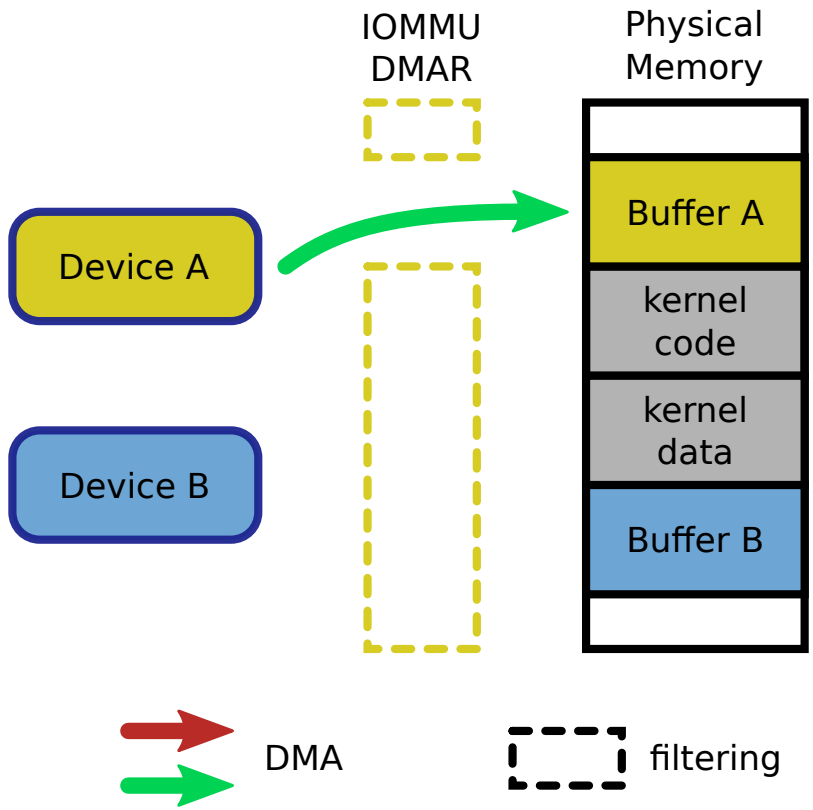
- DMA Remapping (**DMAR**) : Peripheral address space virtualization thanks to translation and filtering
- Interrupt Remapping

DMAR access control



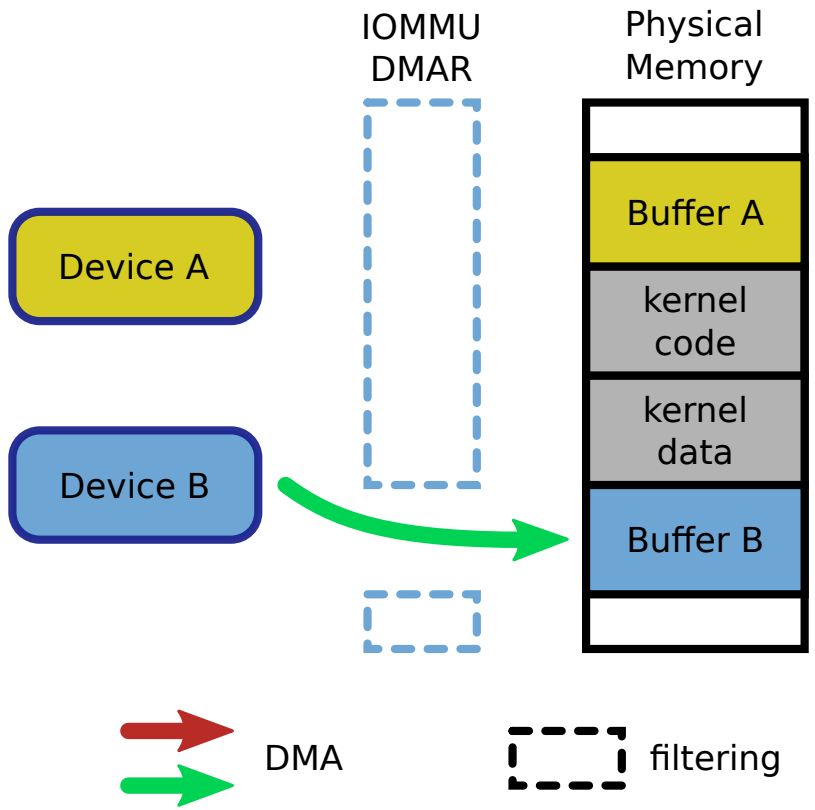
- Multiple memory protection domain
- Unlimited configuration possibilities

DMAR access control



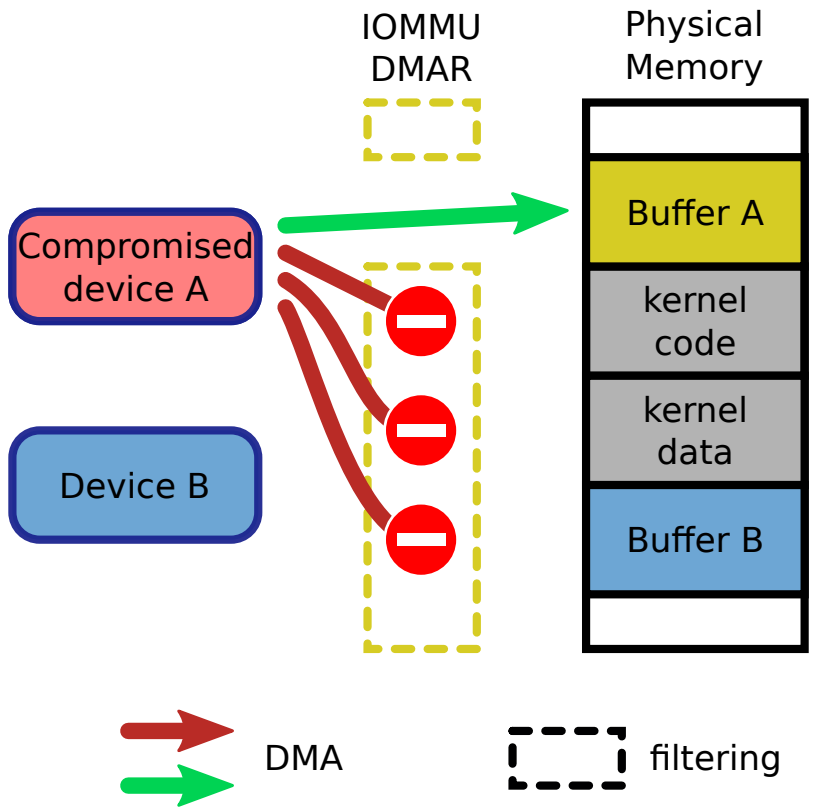
- Multiple memory protection domain
- Unlimited configuration possibilities

DMAR access control



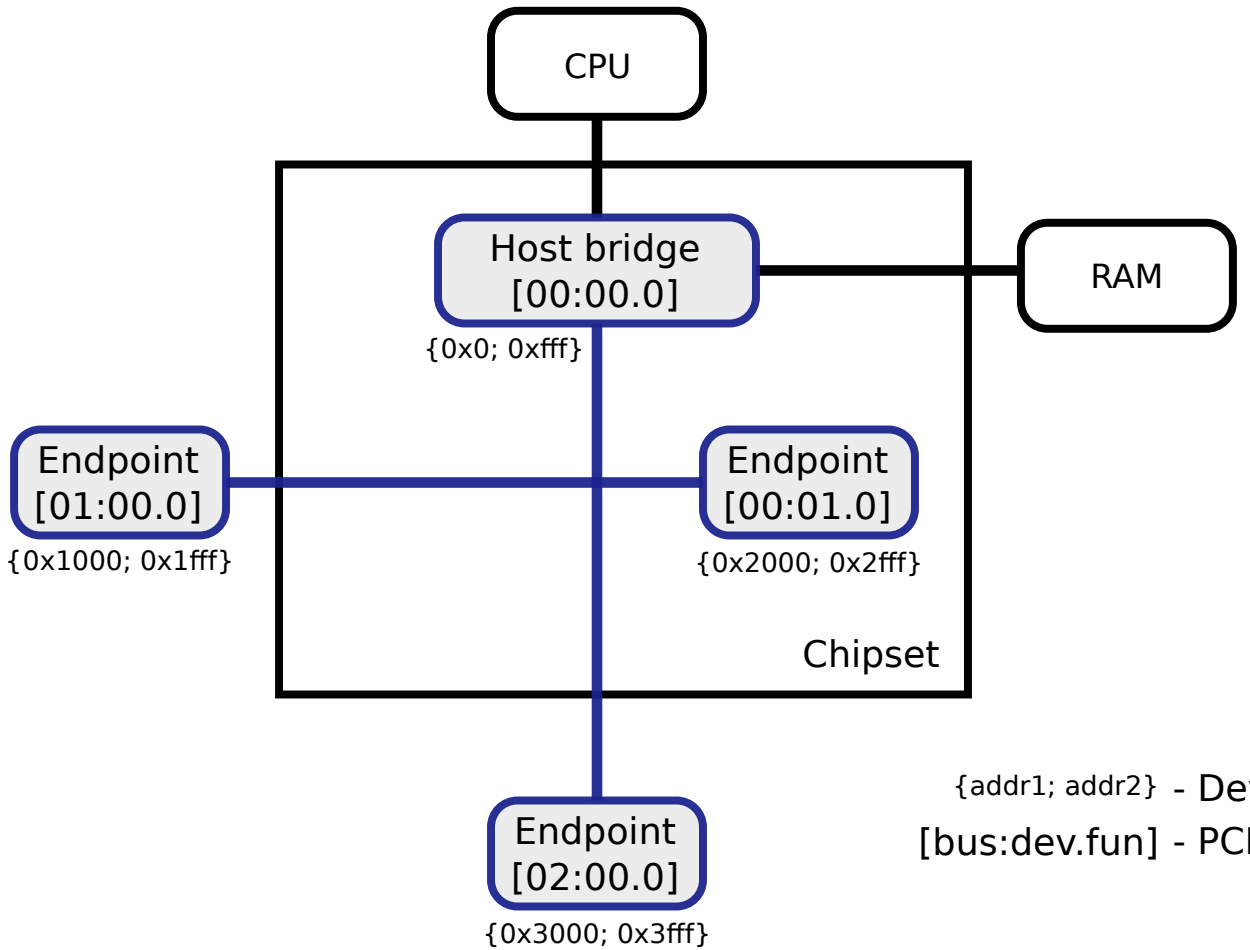
- Multiple memory protection domain
- Unlimited configuration possibilities

DMAR access control



- Multiple memory protection domain
- Unlimited configuration possibilities

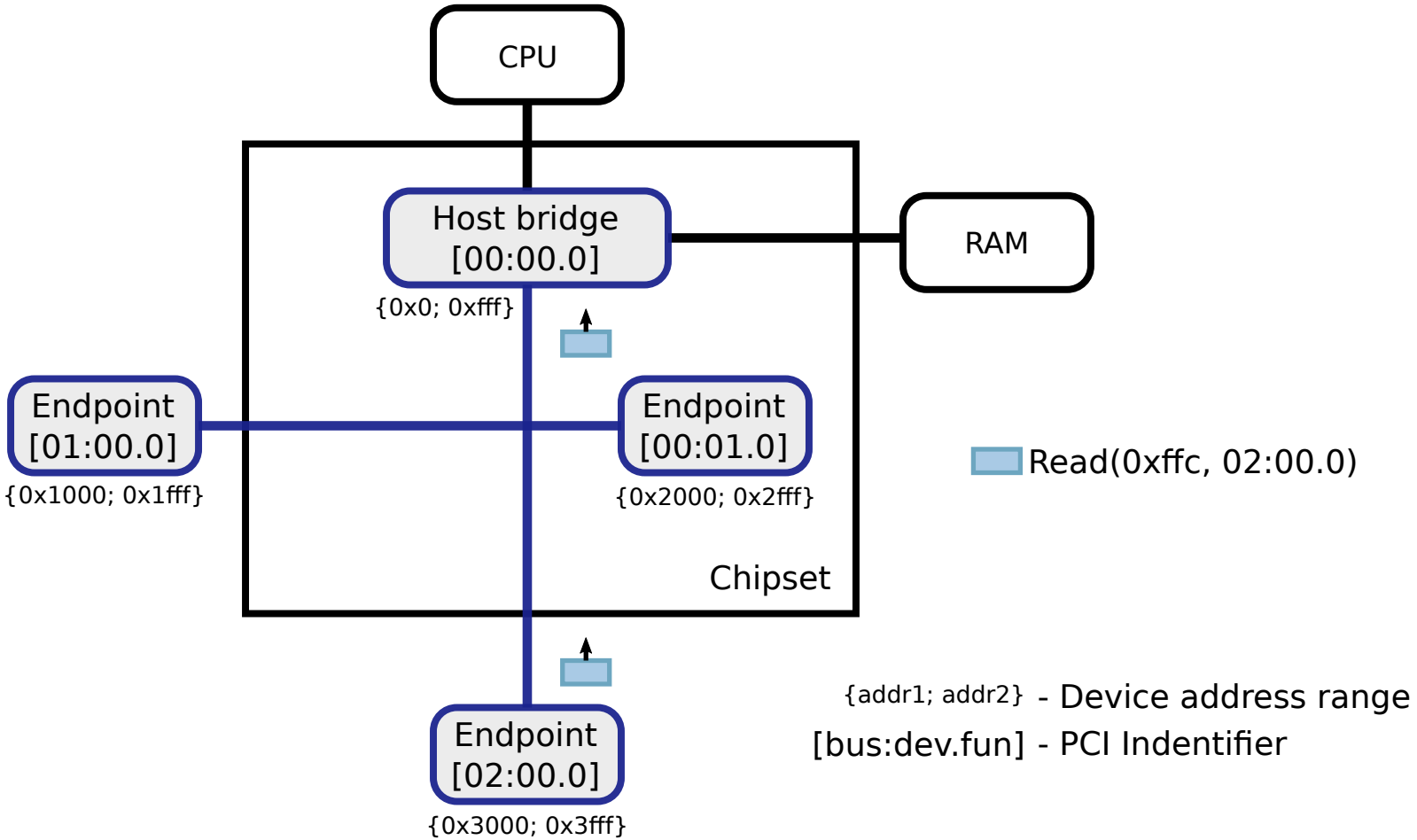
PCI Express bus



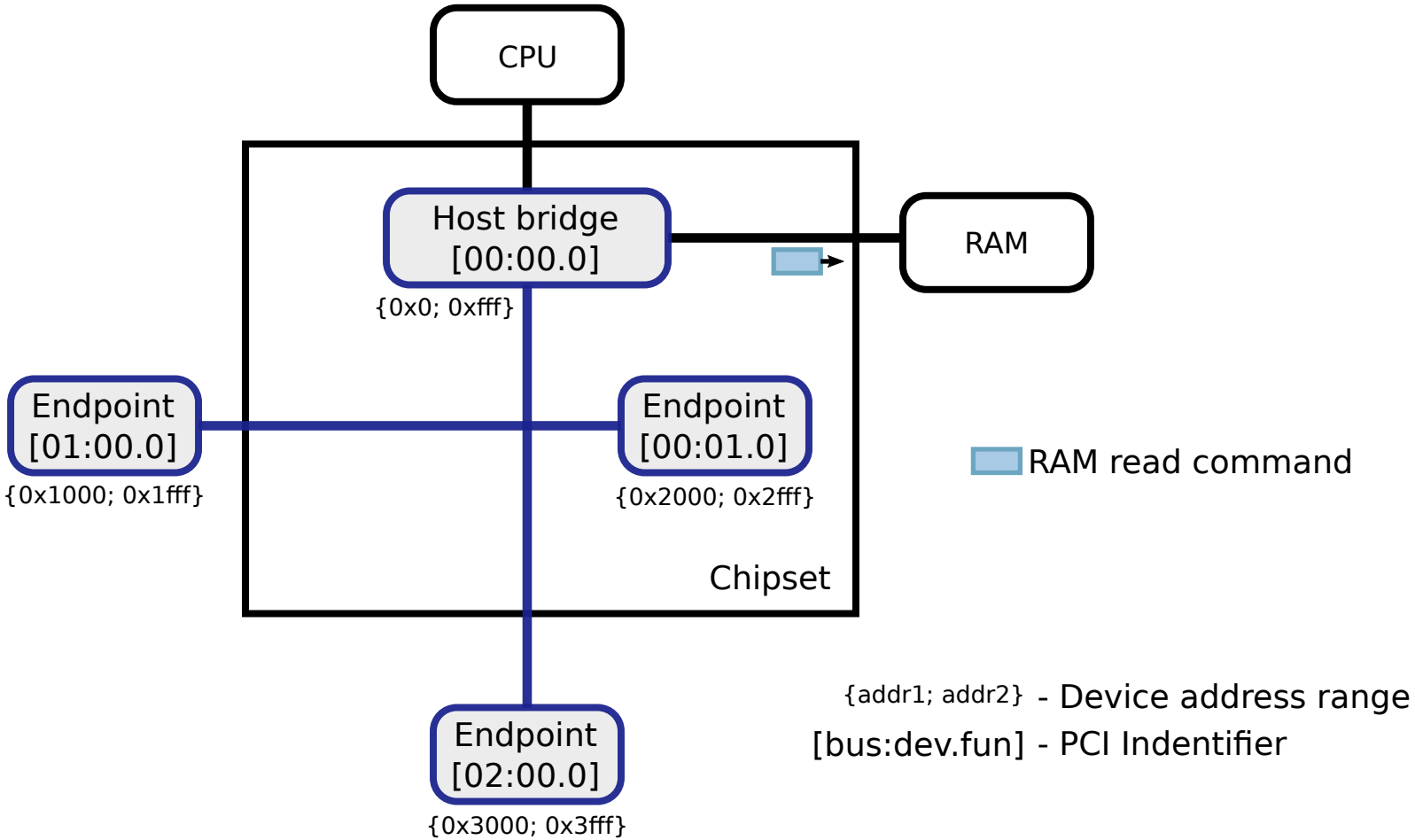
{addr1; addr2} - Device address range

[bus:dev.fun] - PCI Indentifier

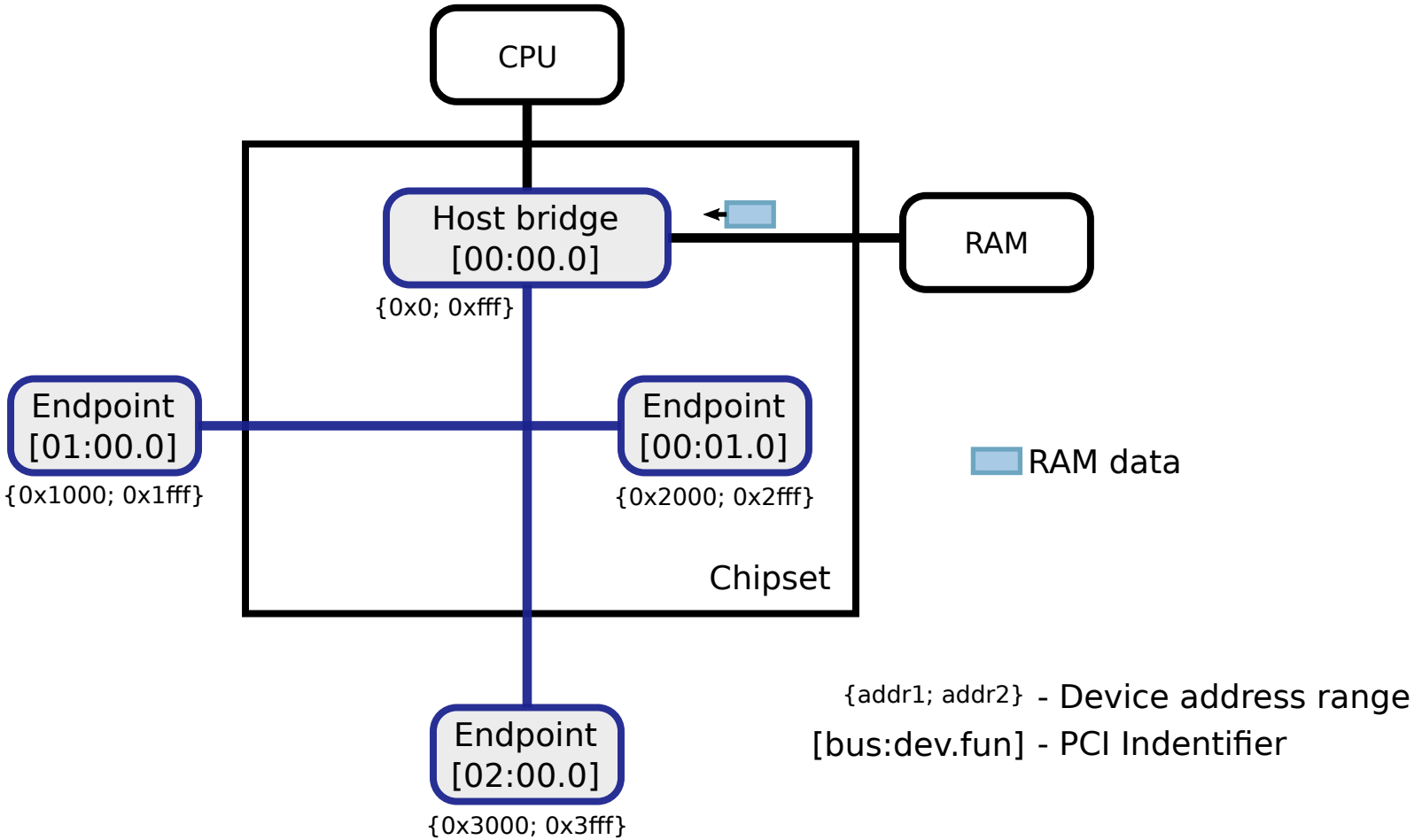
PCI Express Memory Read



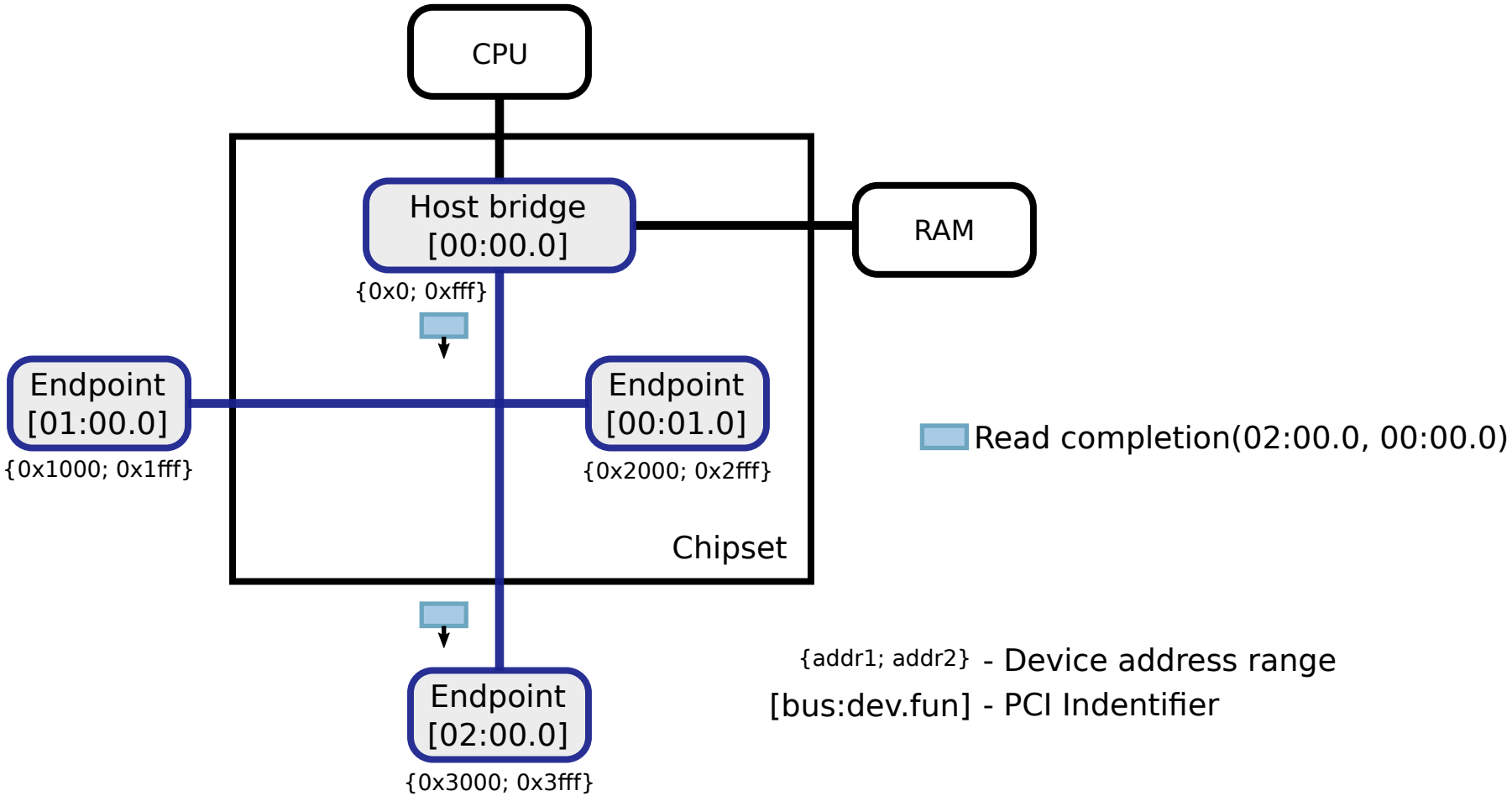
PCI Express Memory Read



PCI Express Memory Read

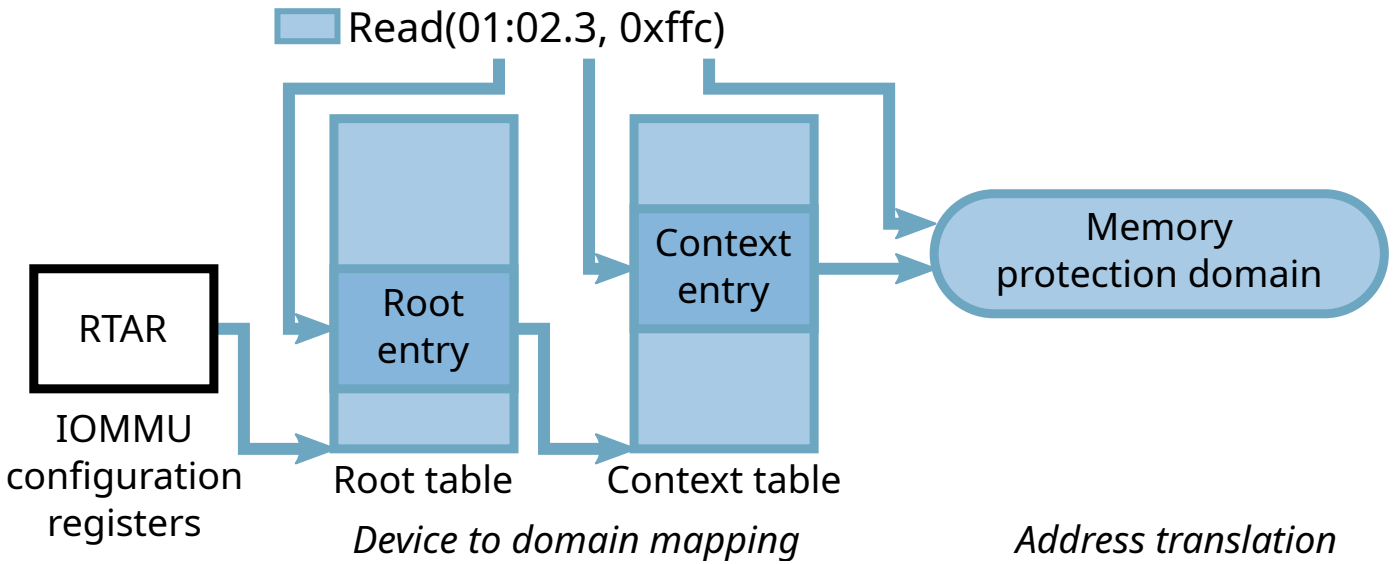


PCI Express Memory Read



DMAR configuration

- A two phases process for translation and filtering with two sets of translation tables
  - Device to domain mapping : identifies the peripheral
  - Address translation : applies access control domain
- Memory structures are placed in RAM and read by the IOMMU located in the host bridge



Outline

1 Background

2 A vulnerability in the IOMMU configuration
- Observations
- Attack

3 Conclusion

Attack goal

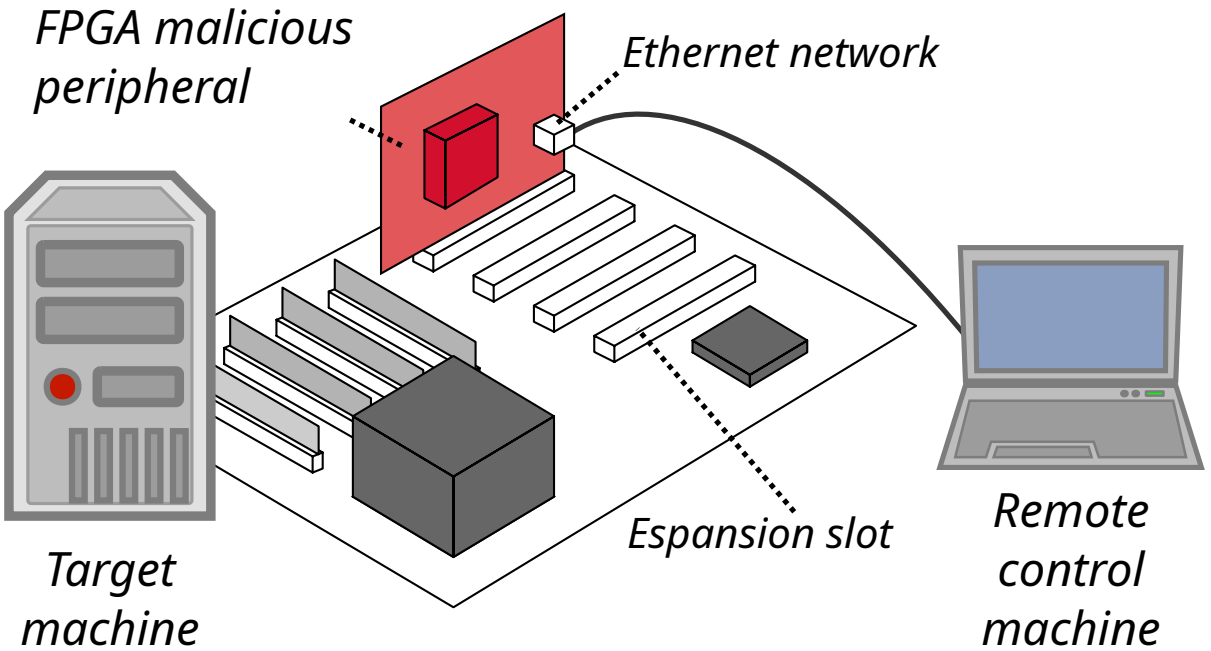
- Keep the DMA rights after the activation of the IOMMU

$\Rightarrow$ Overwrite legitimate IOMMU configuration using DMA

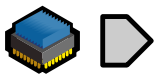
Experimentation setup

- Standard configuration

Linux 4.3.4 (IOMMU Intel)	Intel i7-4770
grub 2.02.beta2	Intel PCH c226



IOMMU vulnerability

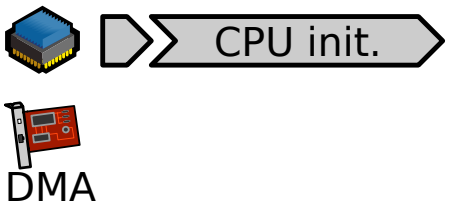


DMA

Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ?

IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ?

IOMMU vulnerability

Our goal : Overwrite legitimate IOMMU configuration using DMA

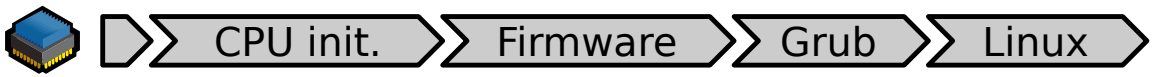
1. When is DMA available ?

IOMMU vulnerability

Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ?

IOMMU vulnerability




DMA

Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ?

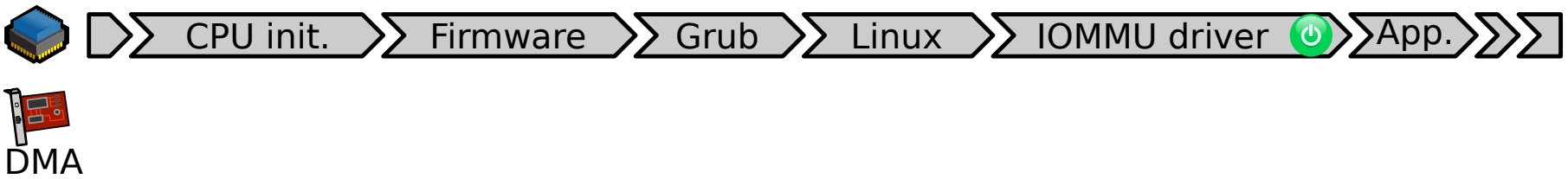
IOMMU vulnerability


DMA

Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ?

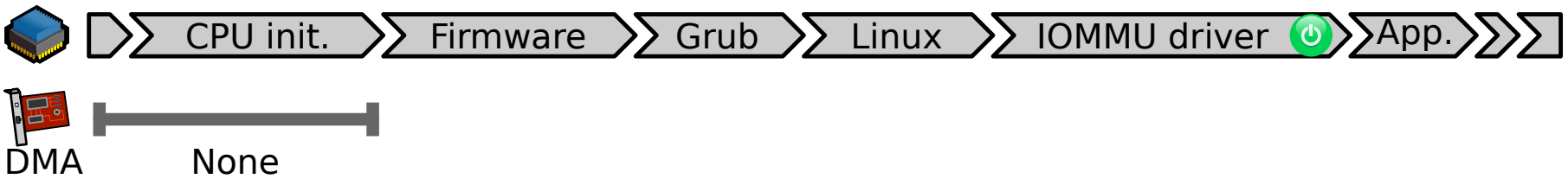
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ?

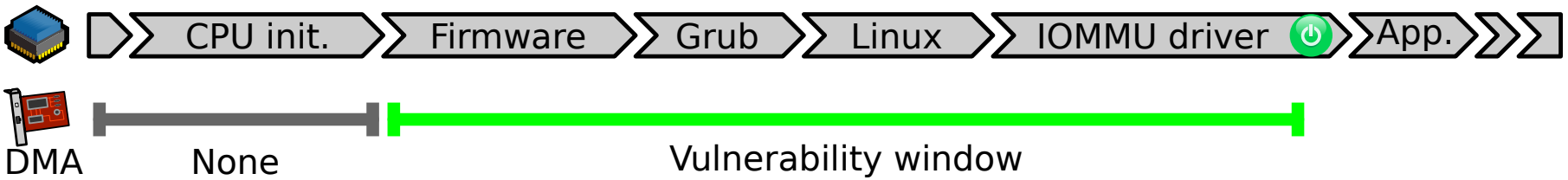
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ?

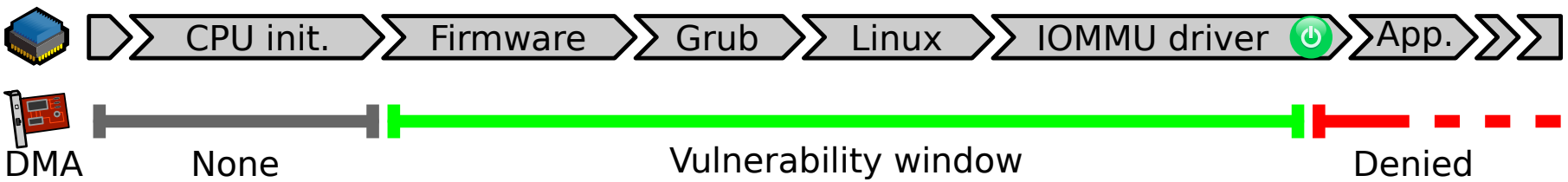
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ? $\Rightarrow$ Vulnerability window

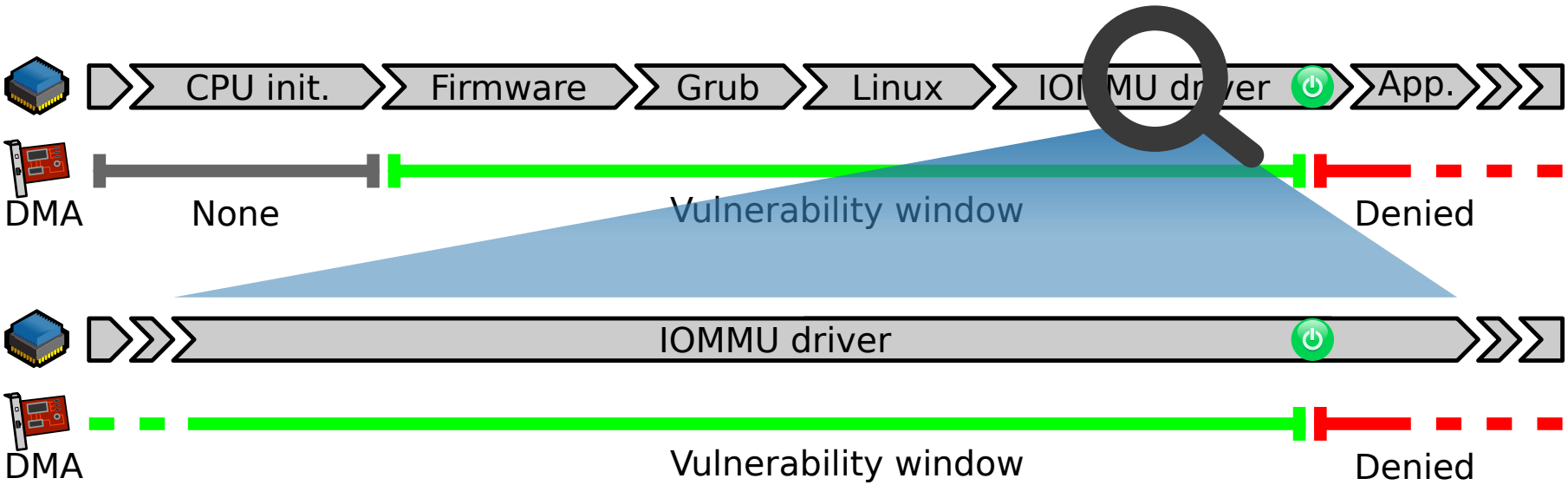
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ? $\Rightarrow$ Vulnerability window
2. When is the configuration actually in memory ?

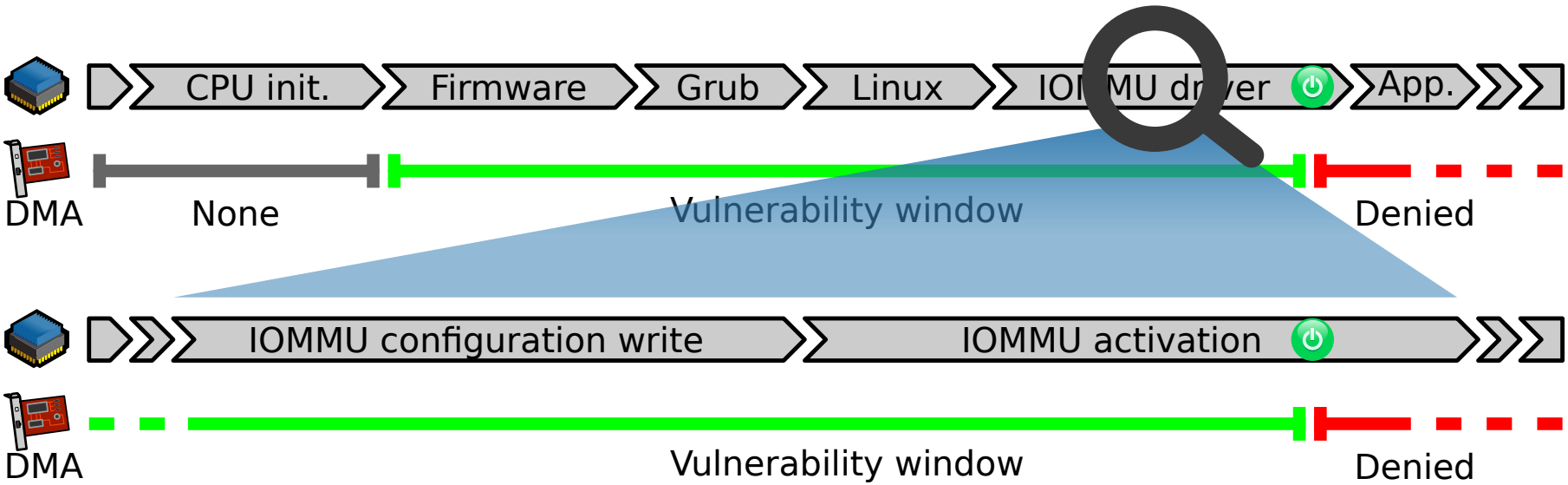
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ? $\Rightarrow$ Vulnerability window
2. When is the configuration actually in memory ?

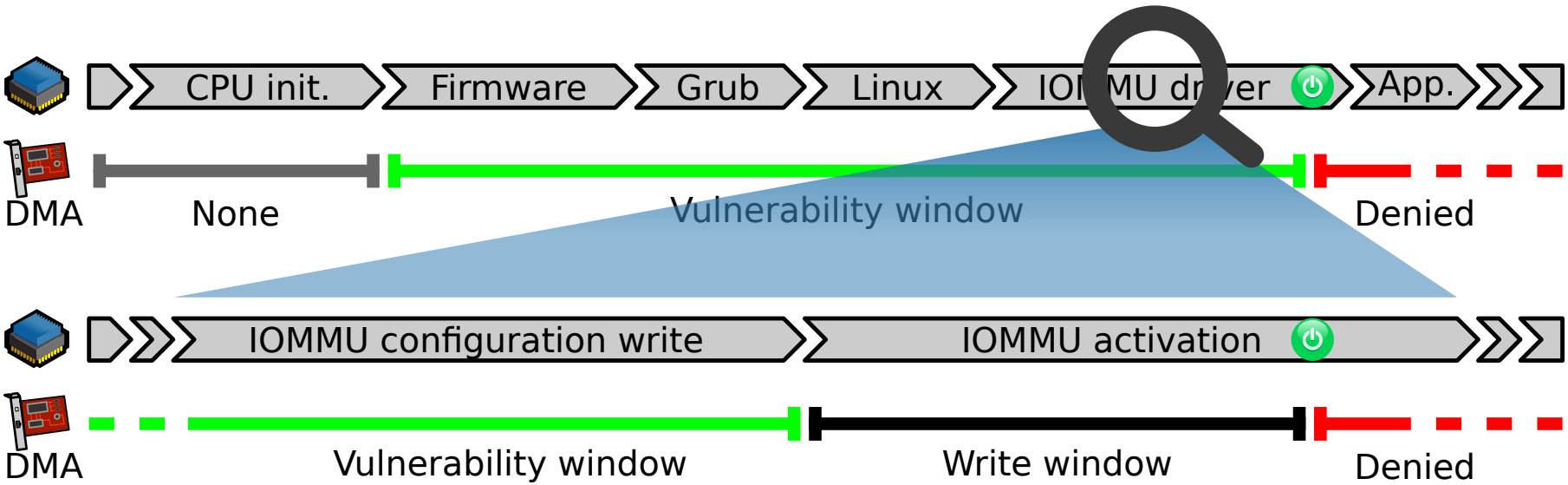
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ? $\Rightarrow$ Vulnerability window
2. When is the configuration actually in memory ?

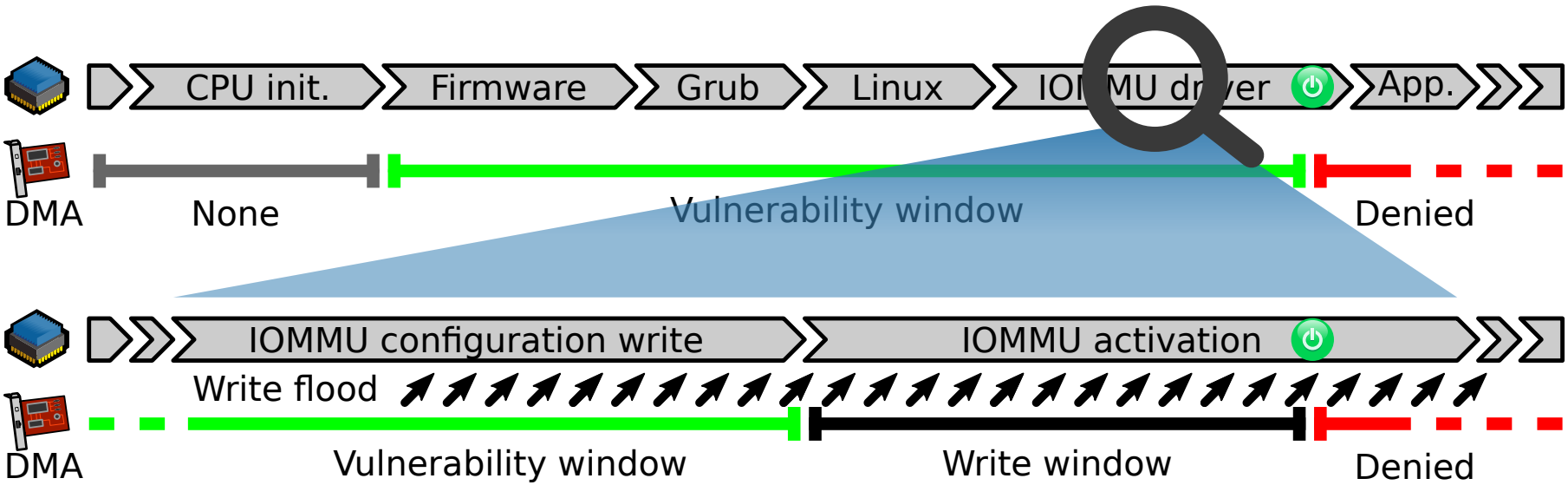
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ? ⇒ Vulnerability window
2. When is the configuration actually in memory ? ⇒ Write window
3. How to be sure not to miss the write window ?

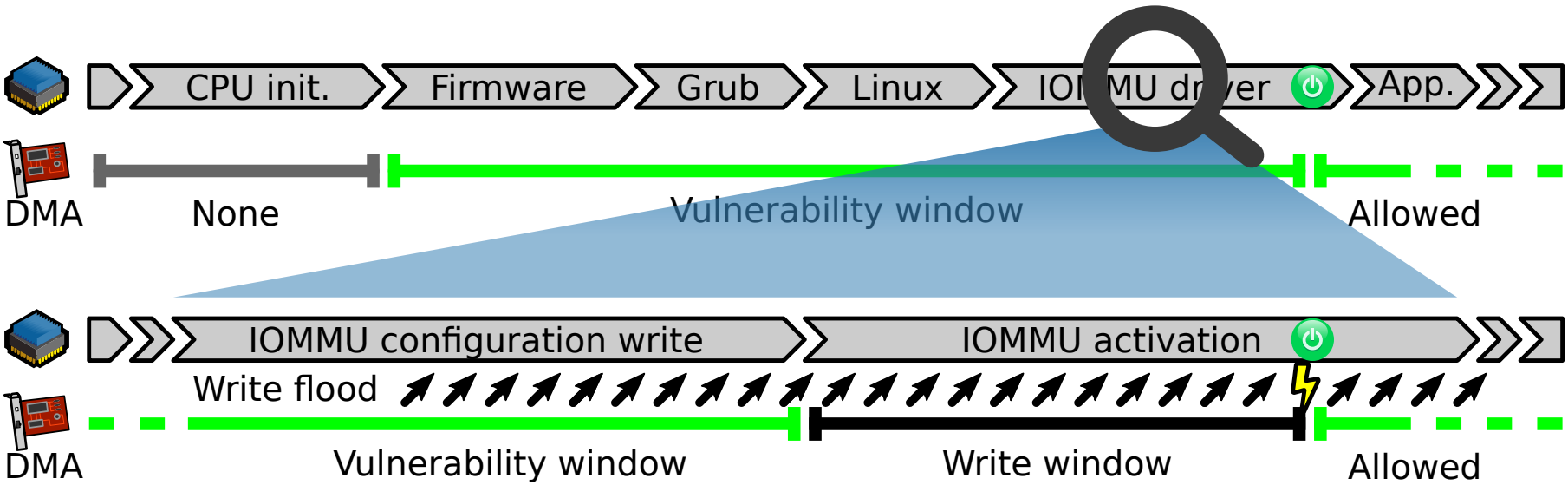
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

1. When is DMA available ? ⇒ Vulnerability window
2. When is the configuration actually in memory ? ⇒ Write window
3. How to be sure not to miss the write window ? ⇒ Write flooding

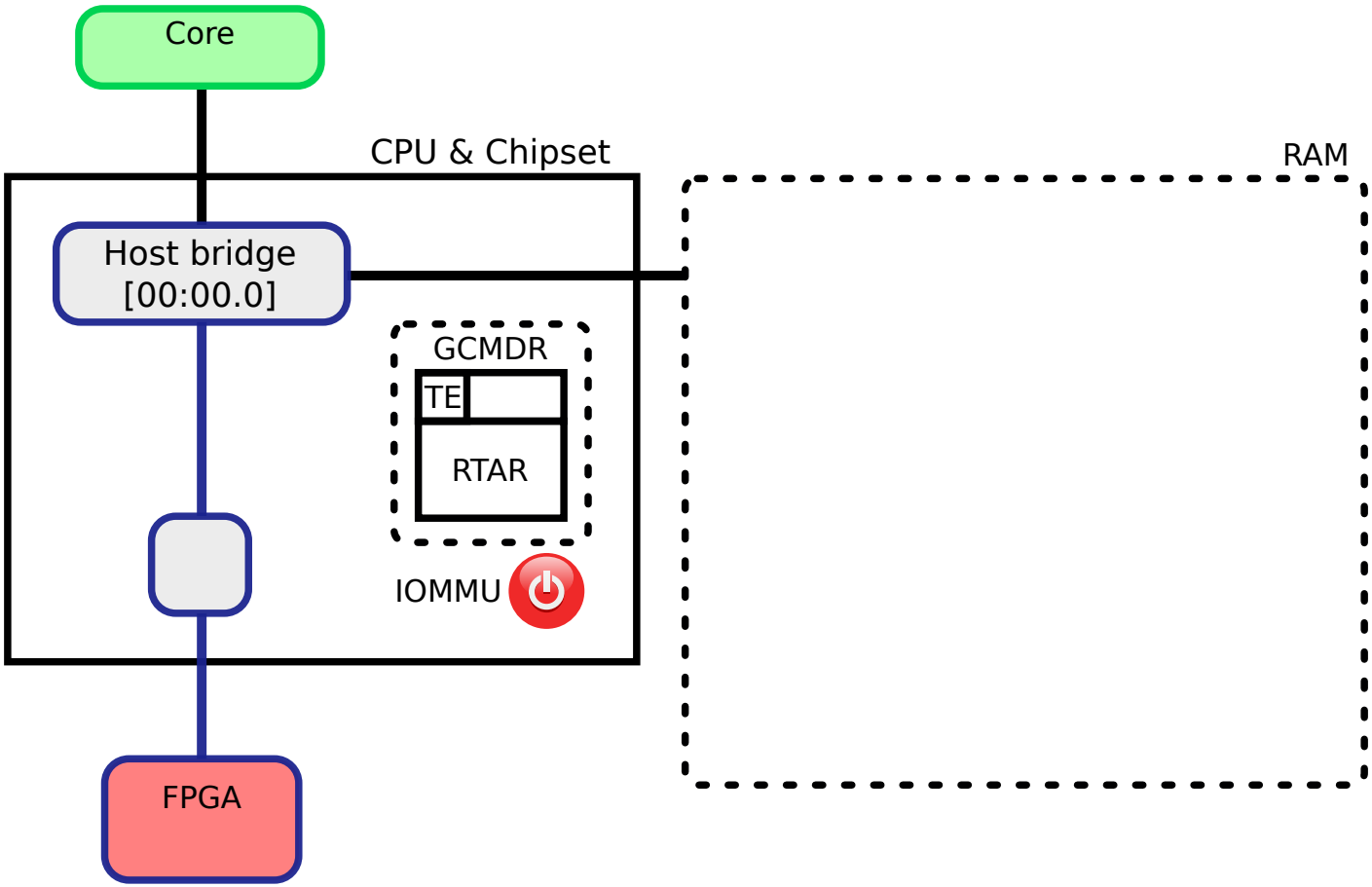
IOMMU vulnerability



Our goal : Overwrite legitimate IOMMU configuration using DMA

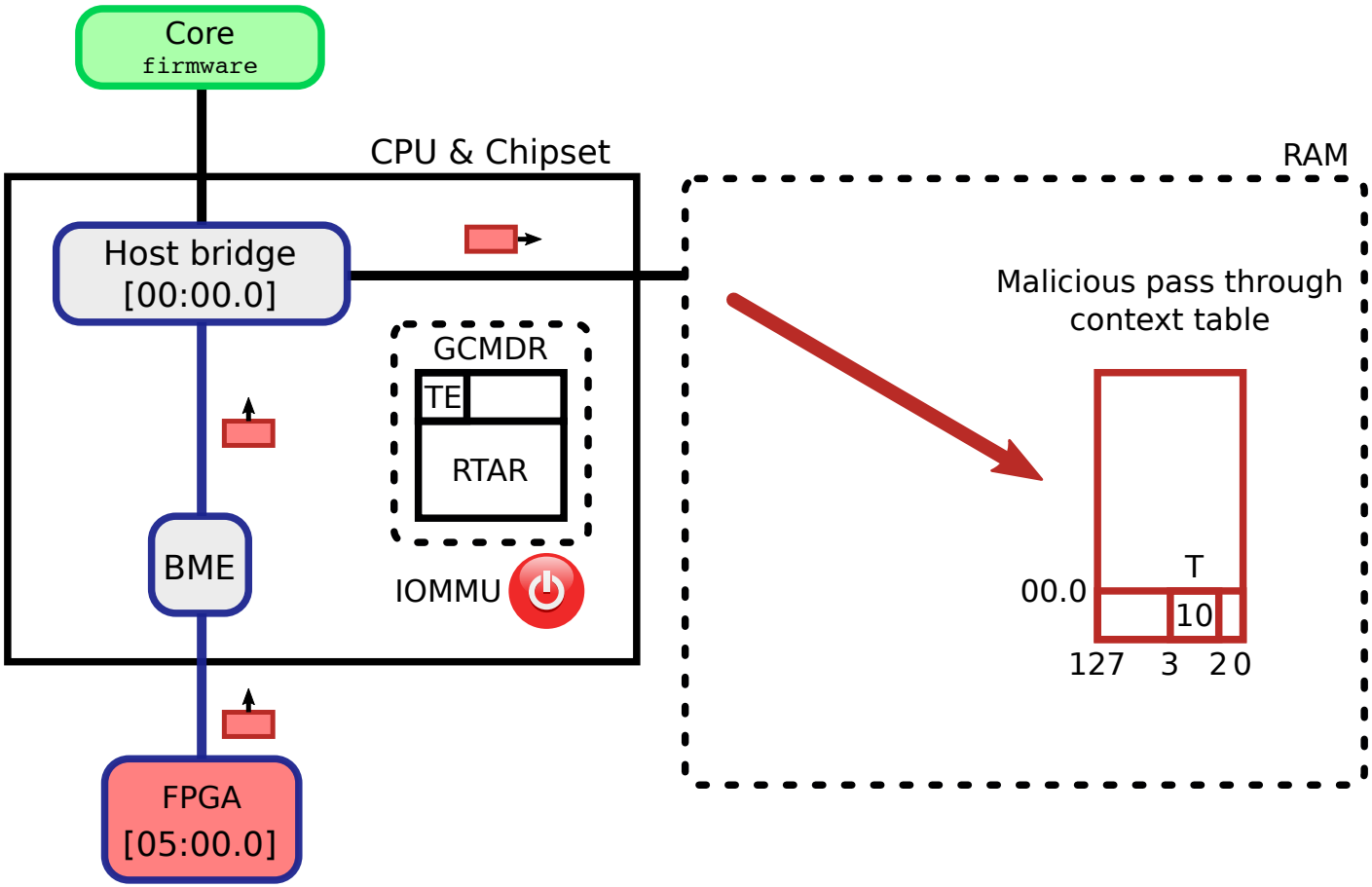
1. When is DMA available ? ⇒ Vulnerability window
2. When is the configuration actually in memory ? ⇒ Write window
3. How to be sure not to miss the write window ? ⇒ Write flooding

Linux driver attack



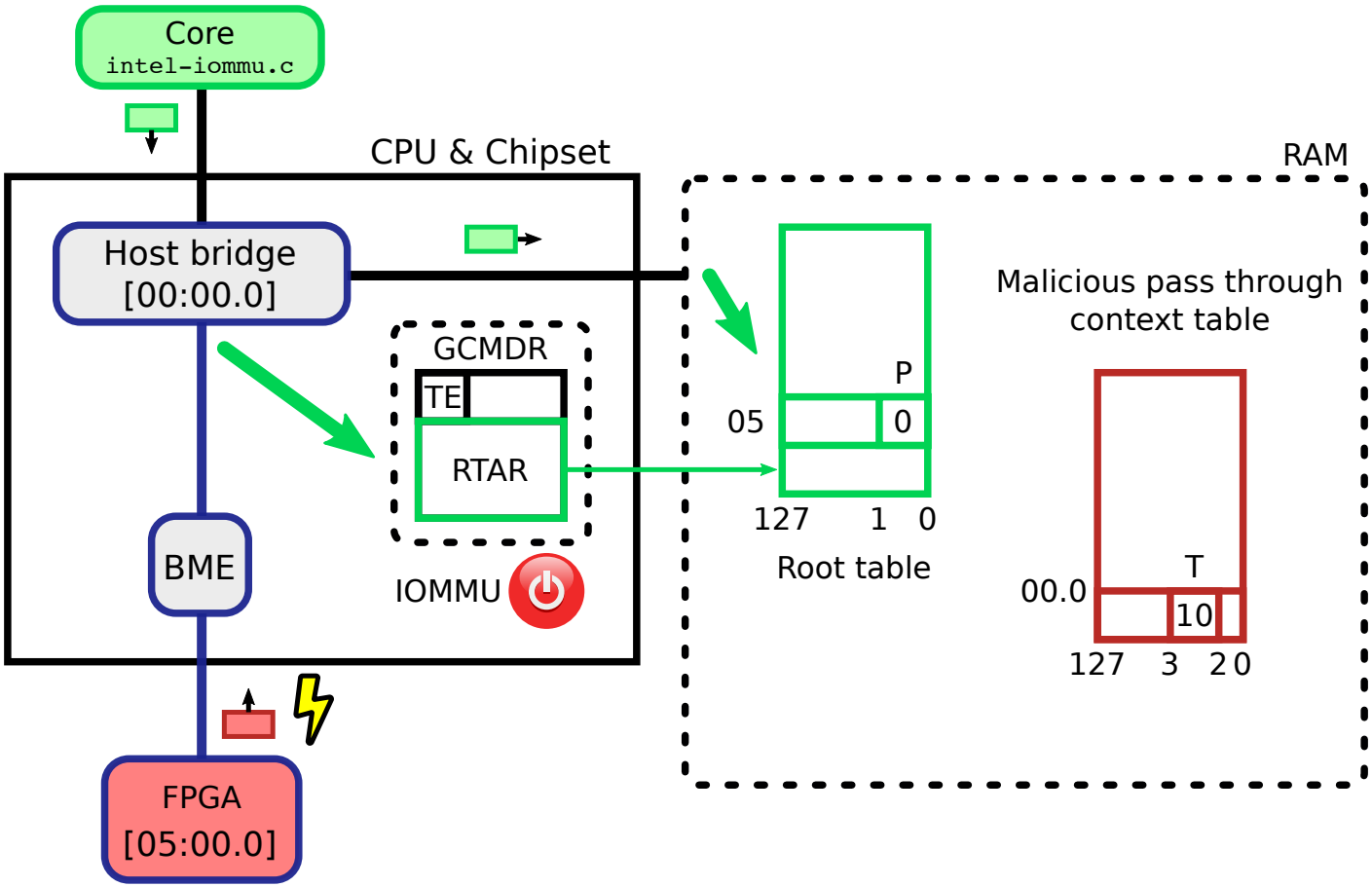
- Platform boot up

Linux driver attack



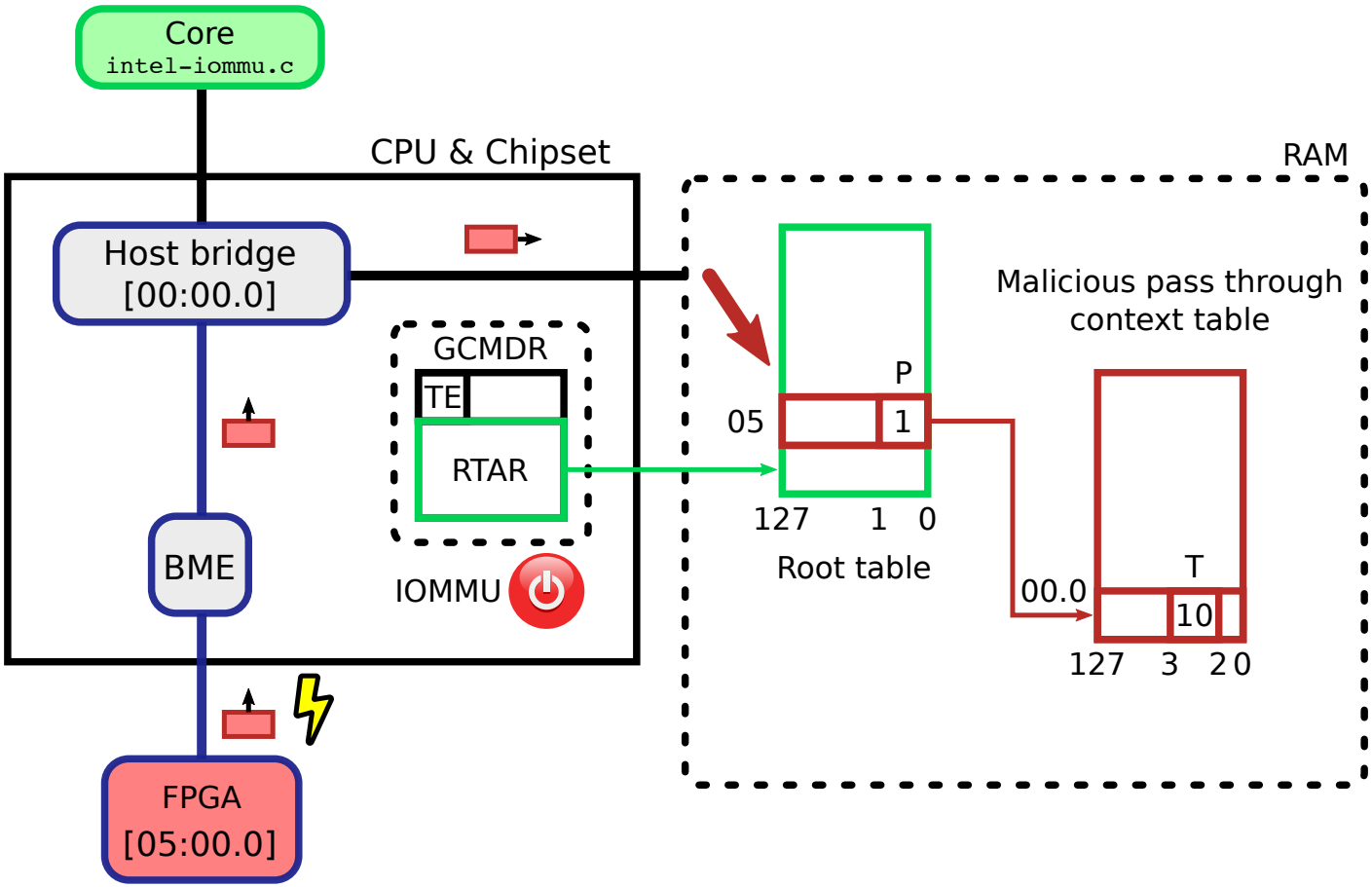
- Copy of a malicious context table in pass through mode

Linux driver attack



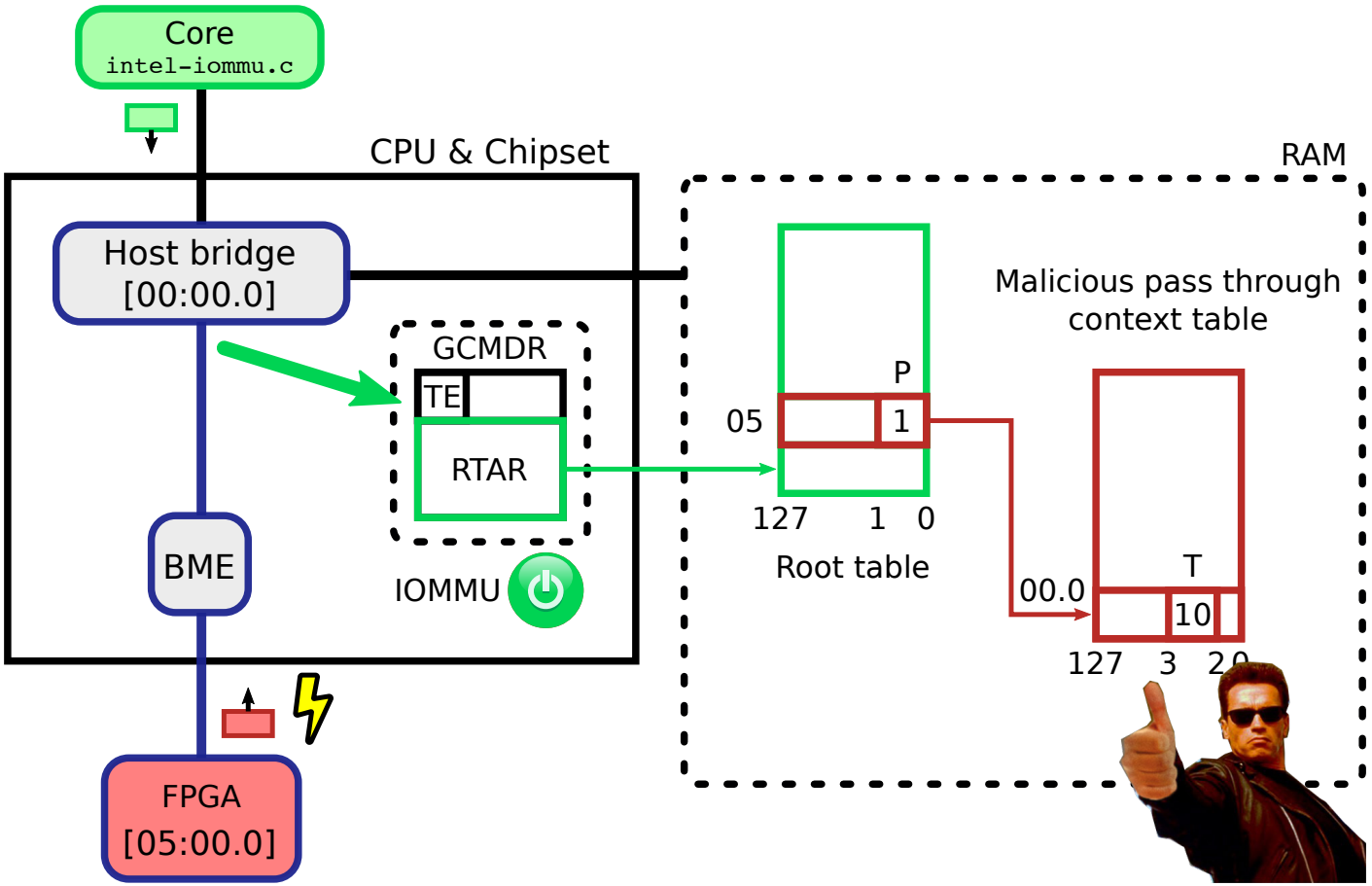
- Legitimate root table configuration by the driver

Linux driver attack



- Overwriting of the Malicious FPGA root entry

Linux driver attack

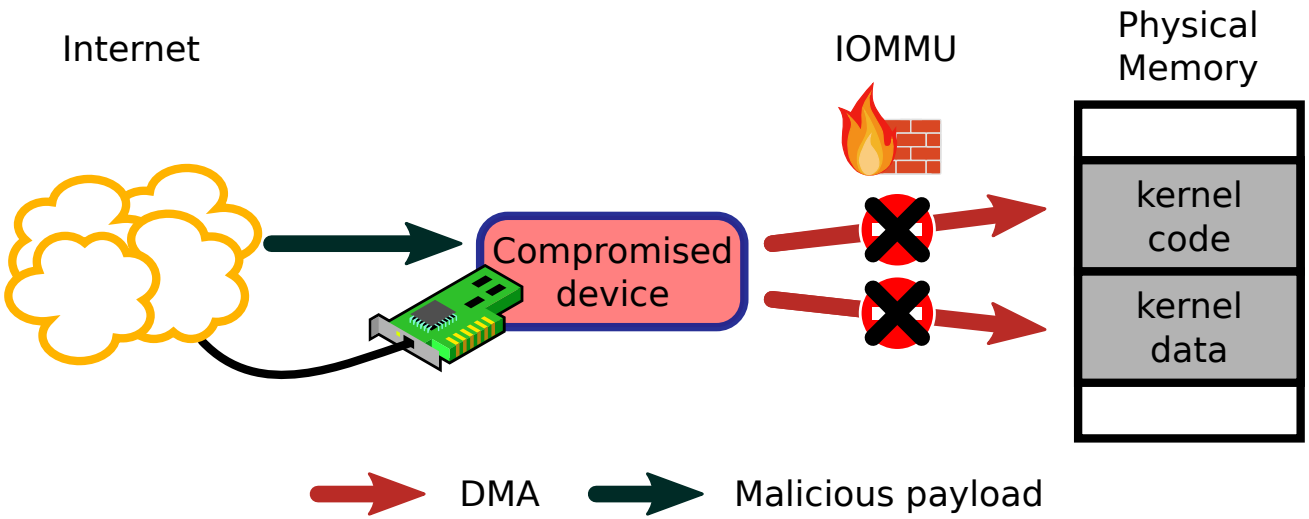


- Legitimate DMAR activation by the linux IOMMU driver

Attack : demonstration

- `http://homepages.laas.fr/nicomett/LADC2016/iommu-pwn-sstic.webm`

Consequences of the attack



⇒ DMA is still available for a remotely controlled compromised device, even with the IOMMU activated

Outline

1 Background

2 A vulnerability in the IOMMU configuration

3 Conclusion

Conclusion et countermeasures

- IOMMUs are not systematically used by operating systems : deactivated by default on linux (Archlinux). Non used on Windows, openbsd, etc.

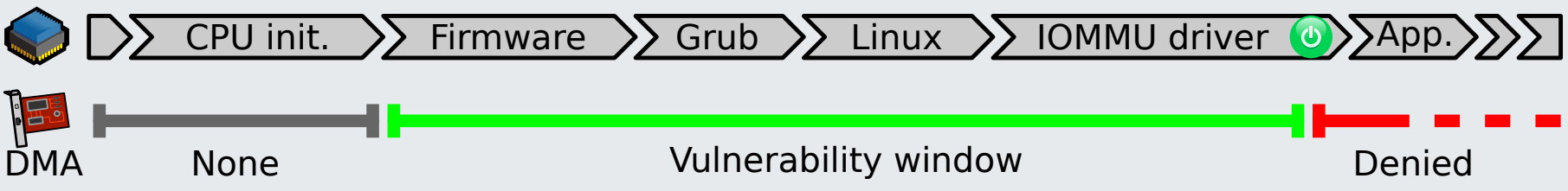
Conclusion et countermeasures

- IOMMUs are not systematically used by operating systems : deactivated by default on linux (Archlinux). Non used on Windows, openbsd, etc.

Conclusion et countermeasures

- IOMMUs are not systematically used by operating systems : deactivated by default on linux (Archlinux). Non used on Windows, openbsd, etc.

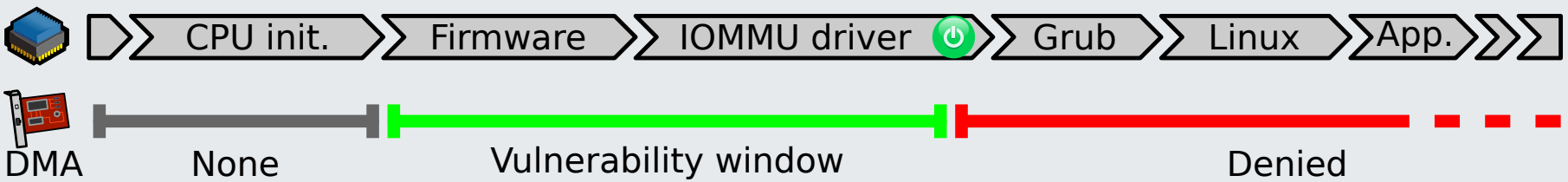
Countermeasures



Conclusion et countermeasures

- IOMMUs are not systematically used by operating systems : deactivated by default on linux (Archlinux). Non used on Windows, openbsd, etc.

Countermeasures

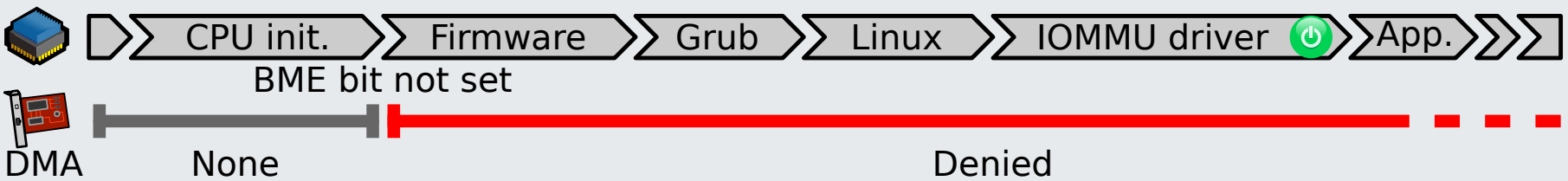


- Change the firmware / OS responsibilities in activating DMAR

Conclusion et countermeasures

- IOMMUs are not systematically used by operating systems : deactivated by default on linux (Archlinux). Non used on Windows, openbsd, etc.

Countermeasures

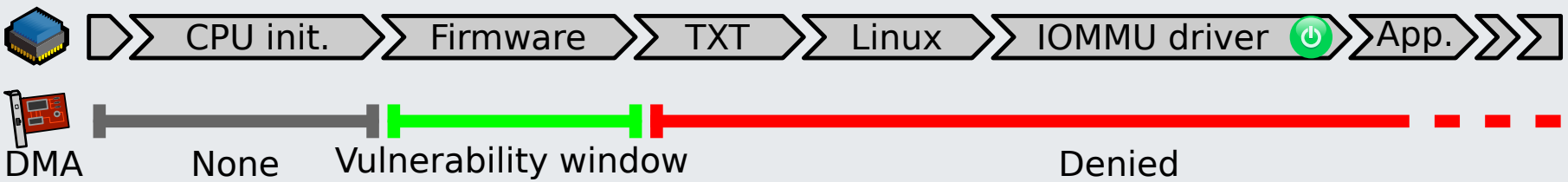


- Change the firmware / OS responsibilities in activating DMAR
- Use PCI to PCI bridges access control (BME bit)

Conclusion et countermeasures

- IOMMUs are not systematically used by operating systems : deactivated by default on linux (Archlinux). Non used on Windows, openbsd, etc.

Countermeasures



- Change the firmware / OS responsibilities in activating DMAR
- Use PCI to PCI bridges access control (BME bit)
- Secured boot with TXT ?

Bypassing IOMMU protection against I/O Attacks

Benoît Morgan, Éric Alata,
Vincent Nicomette and Mohamed Kaâniche

LAAS-CNRS, INSA Toulouse, University of Toulouse

October 19, 2016

Références

[1] Duflot, L., Perez, Y. A., Valadon, G., & Levillain, O. (2010). Can you still trust your network card. CanSecWest/core10, 24-26.