Abusing CPU Hot-Add weaknesses to escalate privileges in Server Datacenters

Cuauhtemoc Chavez-Corona, Jorge Gonzalez-Diaz, Rene Henriquez-Garcia, Laura Fuentes-Castaneda, Jan Seidl

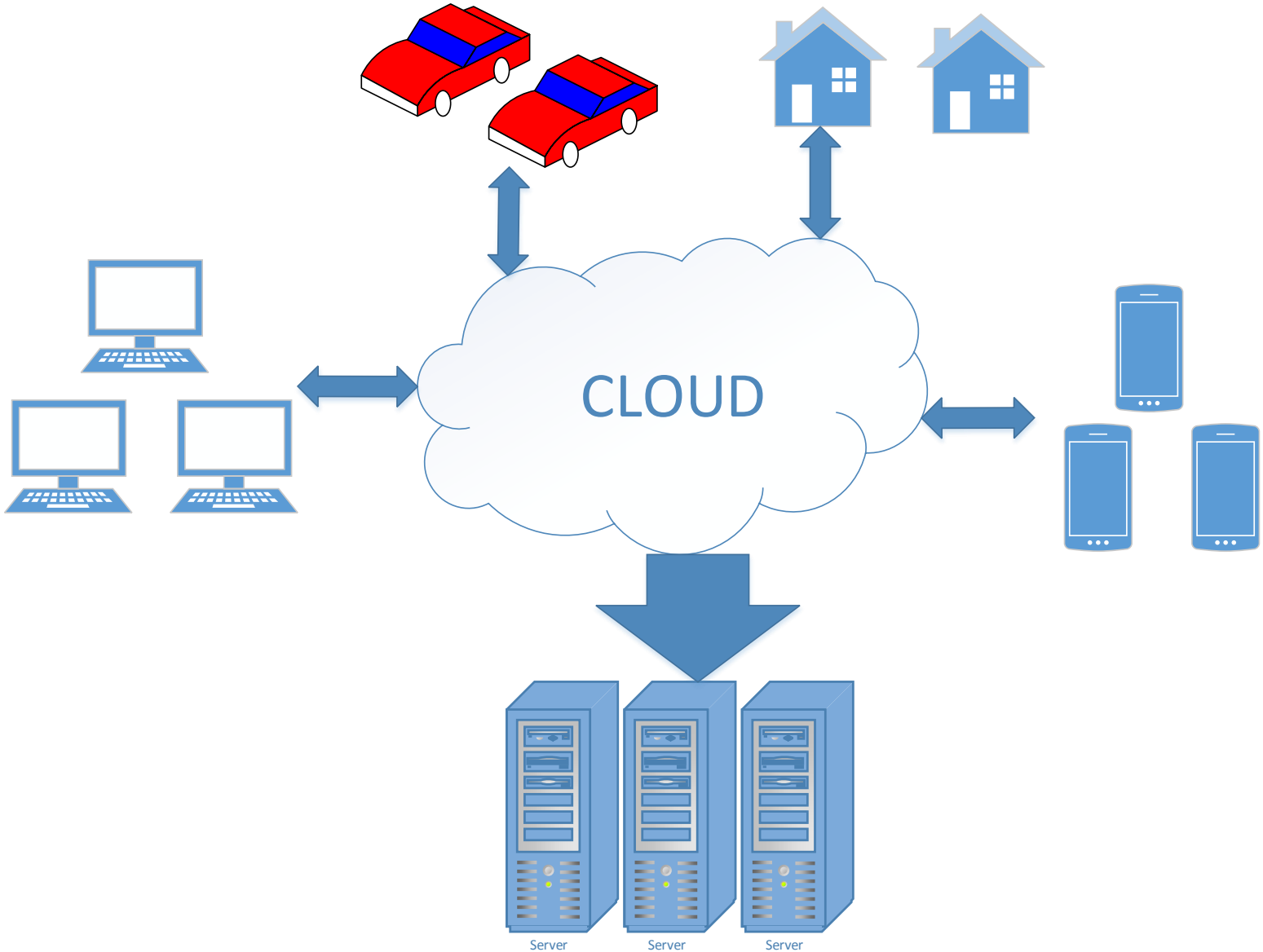
Intel Corporation
Security Center of Excellence
cuauhtemoc.chavez.corona@intel.com
rene.e.henriquez.garcia@intel.com

March 16, 2017

Legal Disclaimer

- The comments and statements are from the authors and not necessarily Intel's
- Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer
- No computer system can be absolutely secure

Background: Datacenter's landscape



Background: Datacenter's landscape

- Mission-critical applications such as e-commerce, ERP, CRM, BI have low tolerance for downtime

Background: Datacenter's landscape

- Mission-critical applications such as e-commerce, ERP, CRM, BI have low tolerance for downtime
- As a response, solutions comprised of robust Hardware + reliable/serviceable FW/SW are continuously being designed

Background: Datacenter's landscape

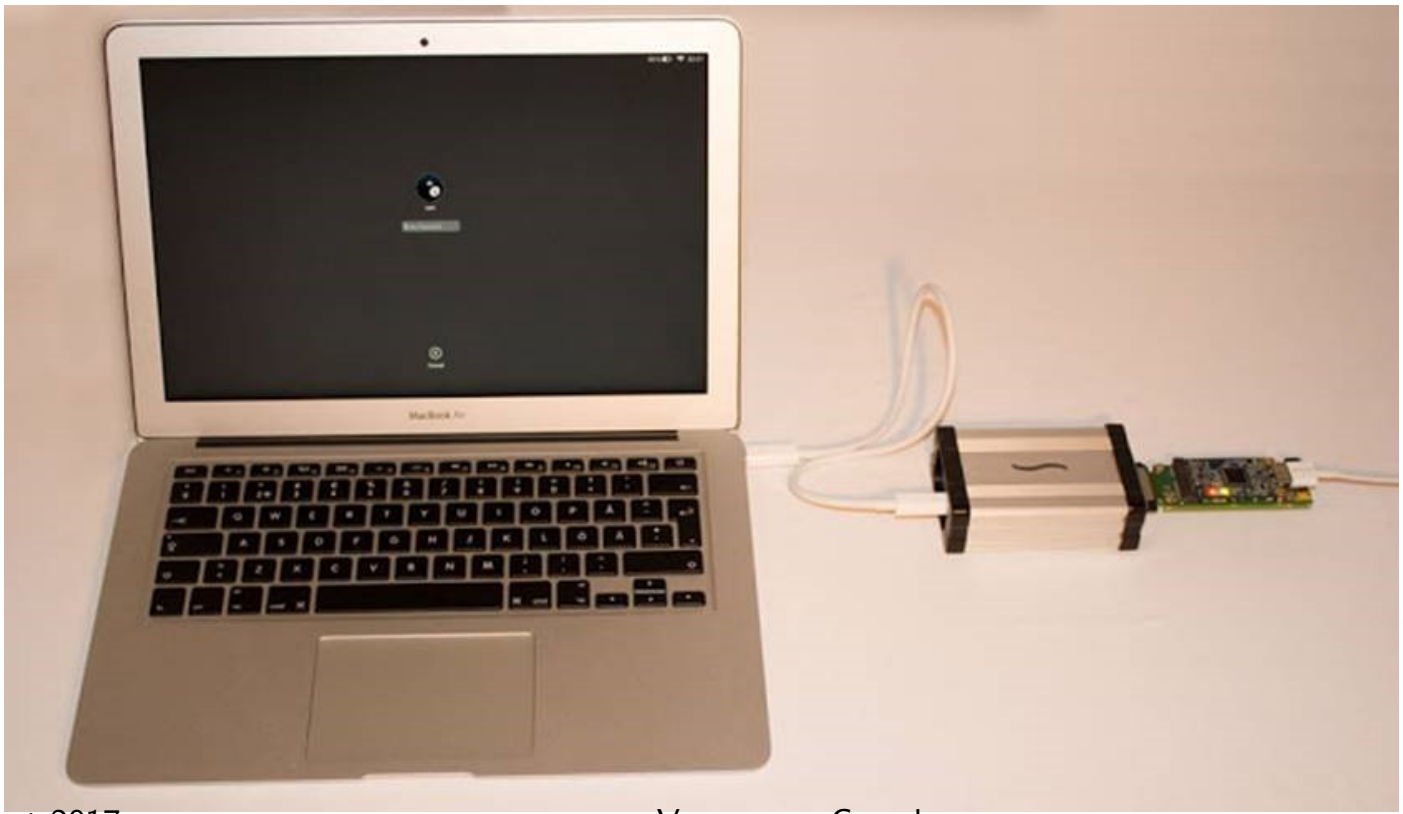
- Mission-critical applications such as e-commerce, ERP, CRM, BI have low tolerance for downtime
- As a response, solutions comprised of robust Hardware + reliable/serviceable FW/SW are continuously being designed
  - Are these new systems being architected such that the attack surface is not increased? We'll see..

Background: Attacks coming from DMA entry point

- Understanding DMA Malware ,Patrick Stewin and Iurii Bystrov, Proceedings of the 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment,2013

Background: Attacks coming from DMA entry point

- Understanding DMA Malware ,Patrick Stewin and Iurii Bystrov, Proceedings of the 9th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment,2013
- Direct Memory Attack the KERNEL, ULF FRISK, DEFCON 24 August 4-7 2016



Definition: RAS features

Reliability

Can be defined as the characteristic that ensures the system will provide correct outputs, and any corrupted data will be detected and repaired.

Availability

Means that the system will be operating during the planned time, avoiding unexpected crashes.

Serviceability

Refers to the simplicity and speed of maintenance and reparation.

Definition: CPU Hot Add

CPU Hot Add (aka CPU on-lining) is a **RAS** feature that allows customers to increase computing power in a Server by adding a new socket to the already running system at Intel® QPI interface without the necessity of shutting down the machine.

Definition: CPU Hot Add

CPU Hot Add (aka CPU on-lining) is a **RAS** feature that allows customers to increase computing power in a Server by adding a new socket to the already running system at Intel® QPI interface without the necessity of shutting down the machine.

- In a multi-CPU system comprised of n processors, one can therefore choose to boot with m CPUs where $m < n$

Definition: CPU Hot Add

CPU Hot Add (aka CPU on-lining) is a **RAS** feature that allows customers to increase computing power in a Server by adding a new socket to the already running system at Intel® QPI interface without the necessity of shutting down the machine.

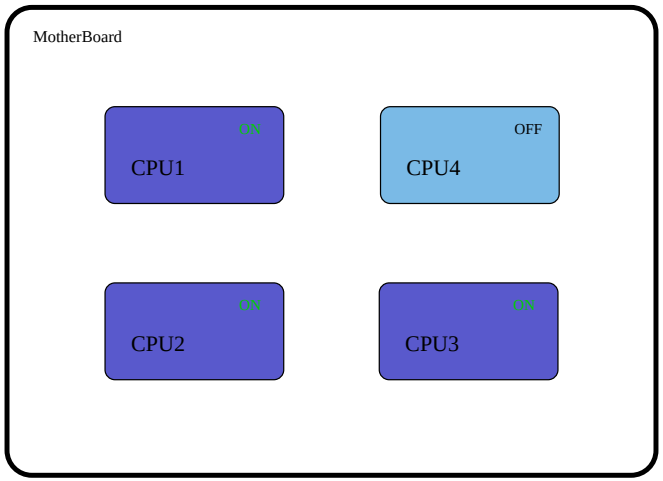
- In a multi-CPU system comprised of n processors, one can therefore choose to boot with m CPUs where $m < n$
- This allows the possibility to increase the computing power later if required by bringing up new CPUs to the already running system

Definition: CPU Hot Add

CPU On-lining requires coordinated support from the complete application stack to ensure correctness while adding a new CPU.

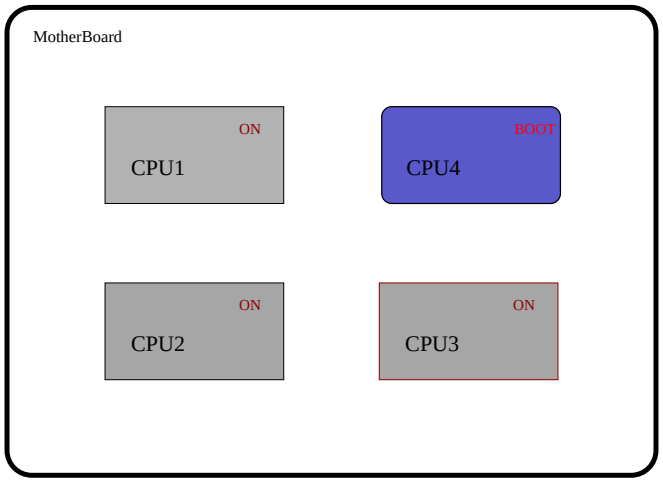
- Hardware. Internal logic in the CPU to drain transactions and prevent originators from sending new ones.
- Firmware. BIOS and SMM routines to trigger, handle and coordinate CPU on-lining.
- Operating System. Currently several OS's support this feature.

High level overview of Hot Add flow



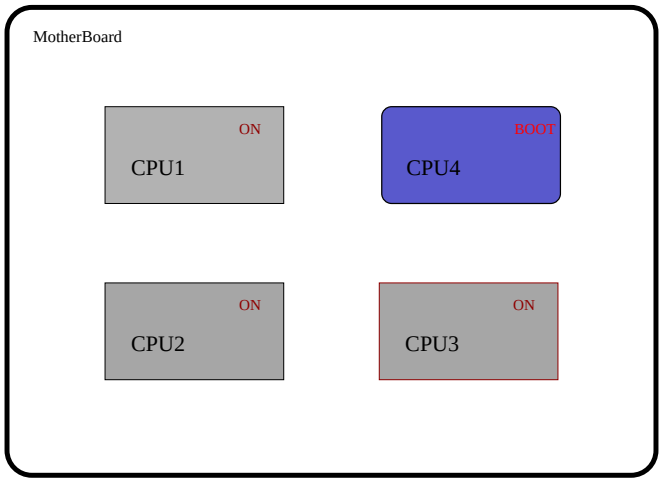
High level overview of Hot Add flow

- Active CPUs enter in quiesce mode



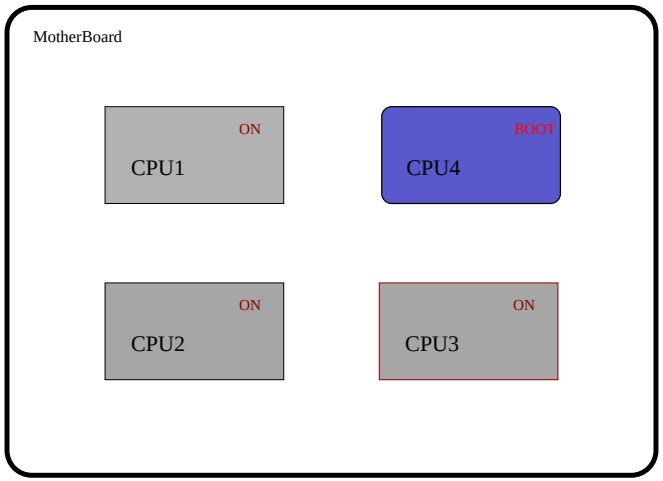
High level overview of Hot Add flow

- Active CPUs enter in quiesce mode
- HotAdd CPU boot process



High level overview of Hot Add flow

- Active CPUs enter in quiesce mode
- HotAdd CPU boot process
  - BSP initialization
  - Memory config
  - etc.



High level overview of Hot Add flow

- Active CPUs enter in quiesce mode
- HotAdd CPU boot process
  - BSP initialization
  - Memory config
  - etc.

Interesting!

1. Boot flow is very sensitive

High level overview of Hot Add flow

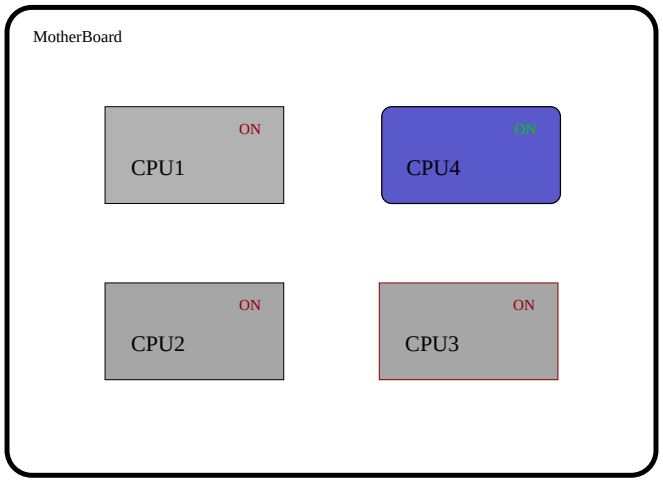
- Active CPUs enter in quiesce mode
- HotAdd CPU boot process
  - BSP initialization
  - Memory config
  - etc.

Interesting!

1. Boot flow is very sensitive
2. Quiesced CPUs need reconfiguration

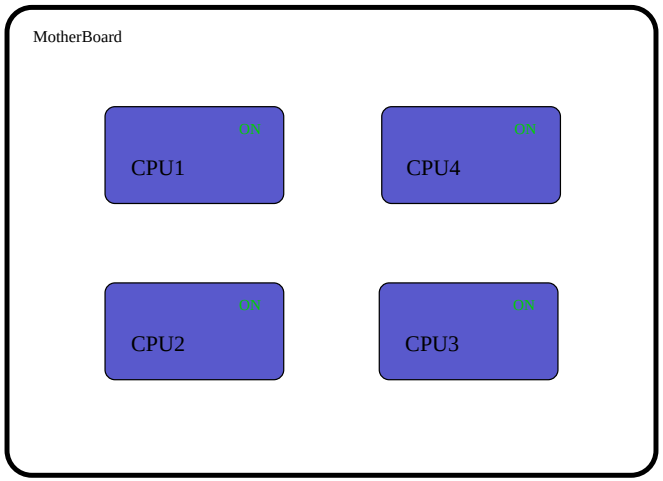
High level overview of Hot Add flow

- Active CPUs enter in quiesce mode
- HotAdd CPU boot process
  - BSP initialization
  - Memory config
  - etc.

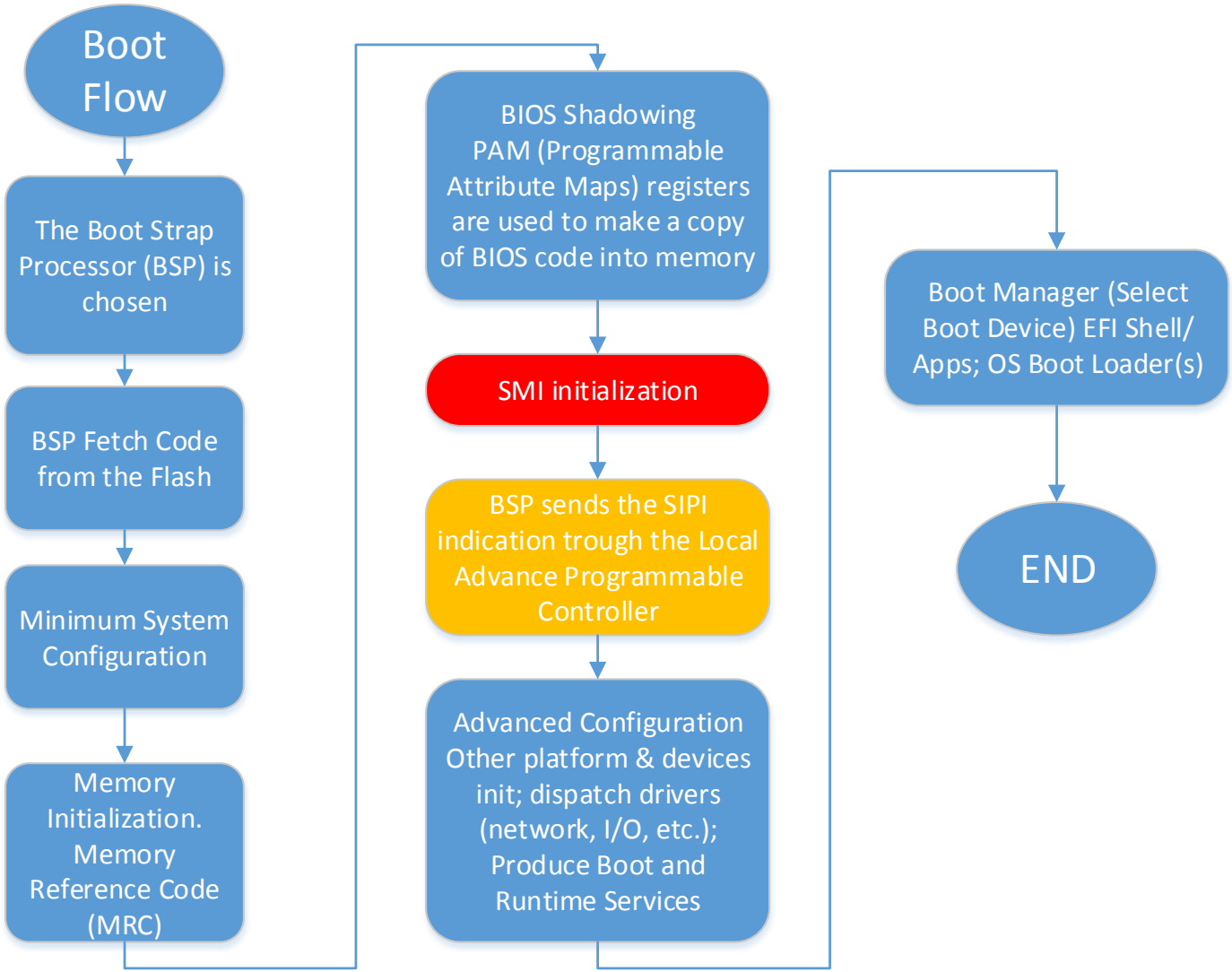


High level overview of Hot Add flow

- Active CPUs enter in quiesce mode
- HotAdd CPU boot process
  - BSP initialization
  - Memory config
  - etc.
- Release quiesced CPUs



Brief overview of Boot Flow



Security Claims from CPU Hot Add definition

One fundamental Security Objective related to CPU Hot Add is that any new CPU to be introduced in the running system must execute a trusted path to ensure its security won't be subverted by any attacker already present in the system

Security Claims from CPU Hot Add definition

One fundamental Security Objective related to CPU Hot Add is that any new CPU to be introduced in the running system must execute a trusted path to ensure its security won't be subverted by any attacker already present in the system

By attackers we mean

- Any rogue code already running in system's CPUs
- DMA agents whose internal FW has been compromised

Assets

There are two interesting regions to be protected in order to ensure security claim presented previously

Assets

There are two interesting regions to be protected in order to ensure security claim presented previously

- 0x38000: Holds the code to be executed in the first SMI by the newly-added CPU in order to perform SMBASE relocation.

Assets

There are two interesting regions to be protected in order to ensure security claim presented previously

- 0x38000: Holds the code to be executed in the first SMI by the newly-added CPU in order to perform SMBASE relocation.
- 0xe2000: Holds SIPI initialization vector code vital for the newly-added CPU and its integration into the running system.

Why do we care about those assets?

- SMM has superior privileges as it can change different settings which cannot be modified by OS

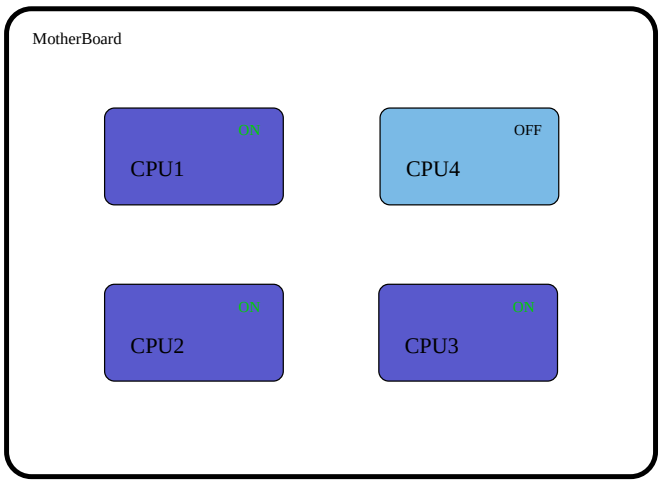
Why do we care about those assets?

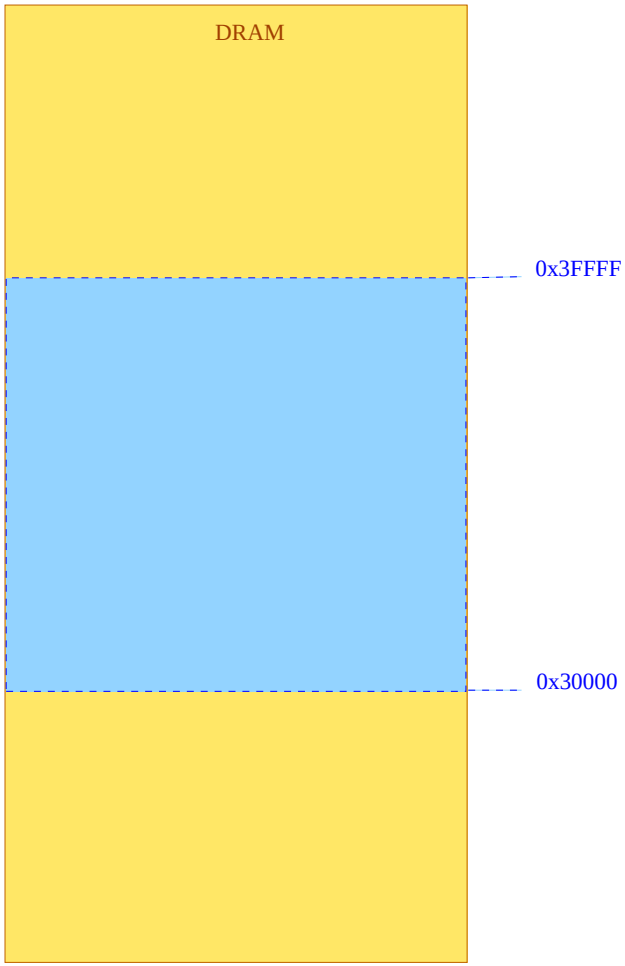
- SMM has superior privileges as it can change different settings which cannot be modified by OS
- In Servers, it is usually referred to as ring -2 whereas OS is being considered as ring 0

Why do we care about those assets?

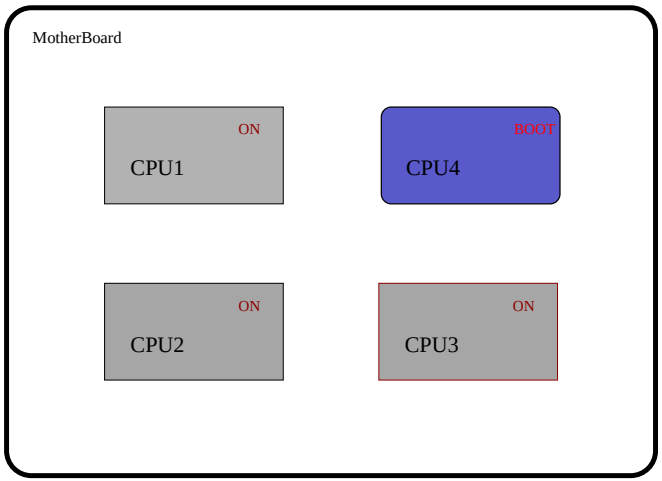
- SMM has superior privileges as it can change different settings which cannot be modified by OS
- In Servers, it is usually referred to as ring -2 whereas OS is being considered as ring 0
- Corrupting Startup Inter-Process Interrupt vector code is also interesting for an attacker as it could potentially be used to misconfigure initial configuration of the newly-added CPU

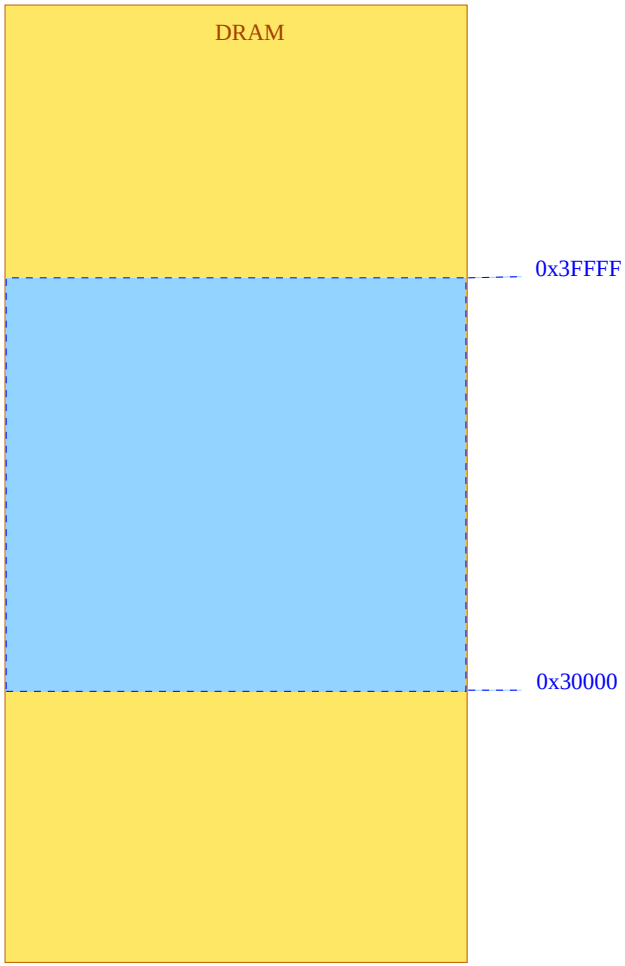
0x38000 attack: Escalate to SMM privileges in a Server



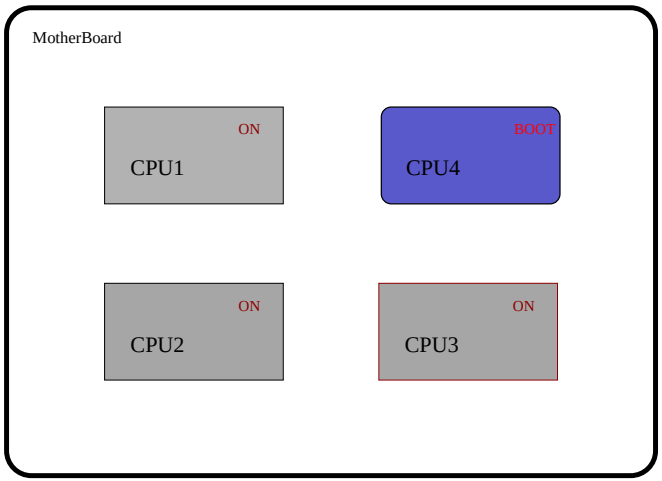


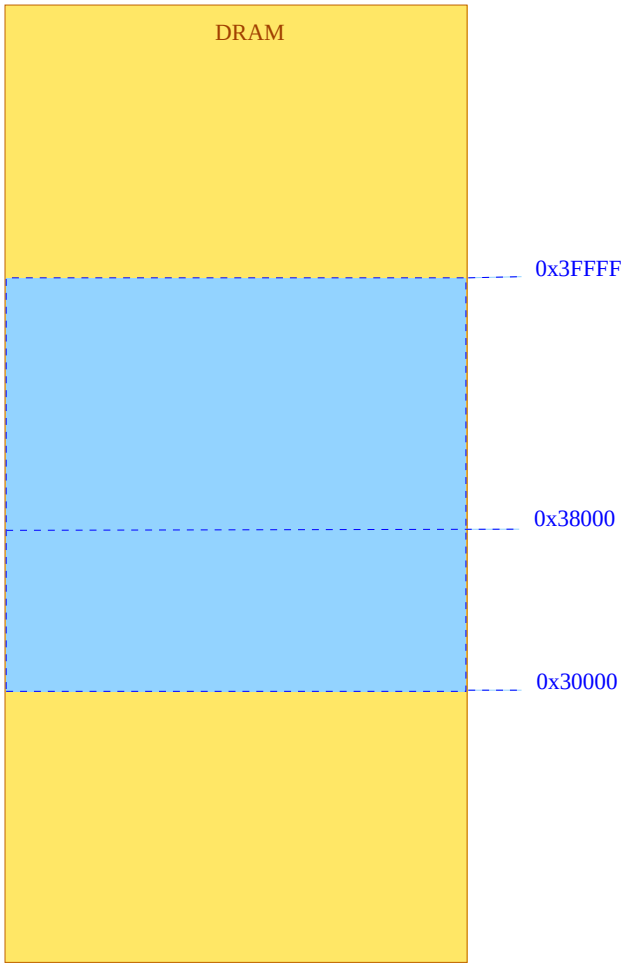
0x38000 attack: Escalate to SMM privileges in a Server



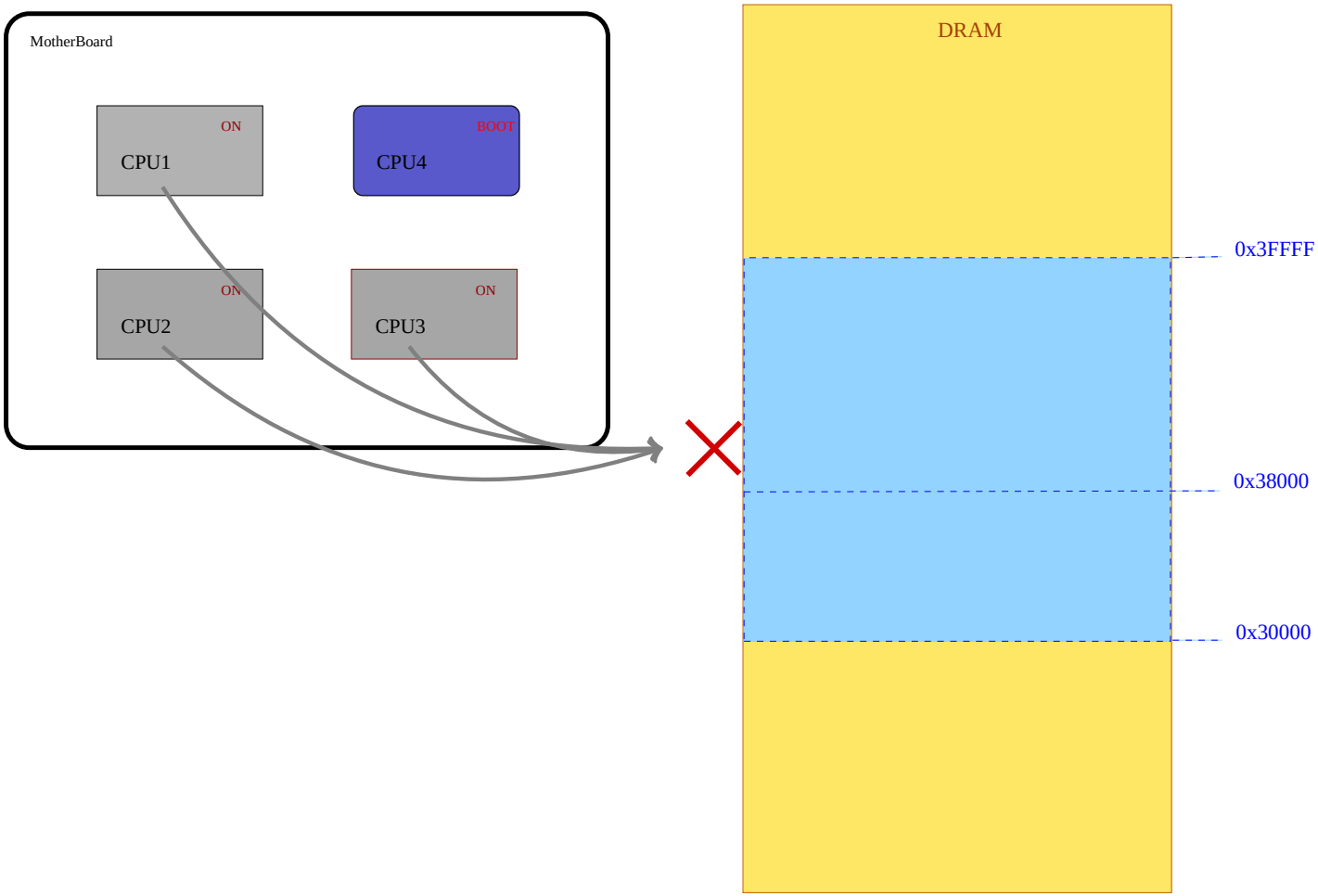


0x38000 attack: Escalate to SMM privileges in a Server

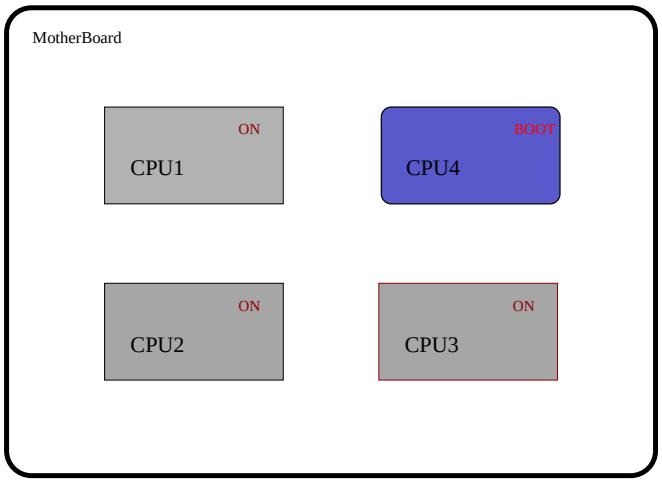


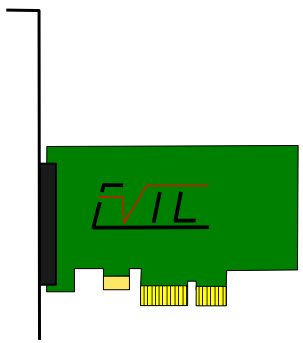


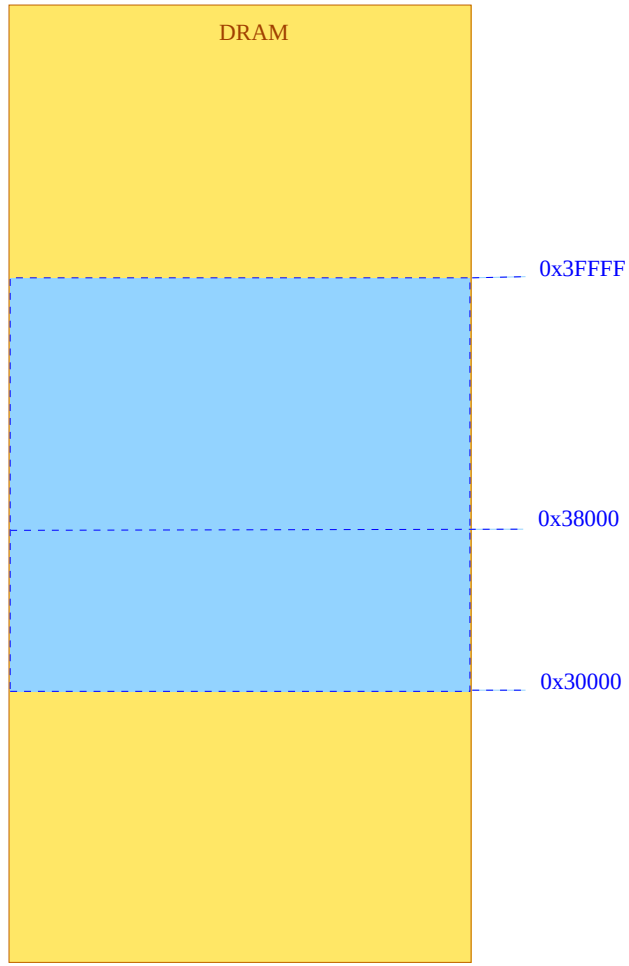
0x38000 attack: Escalate to SMM privileges in a Server



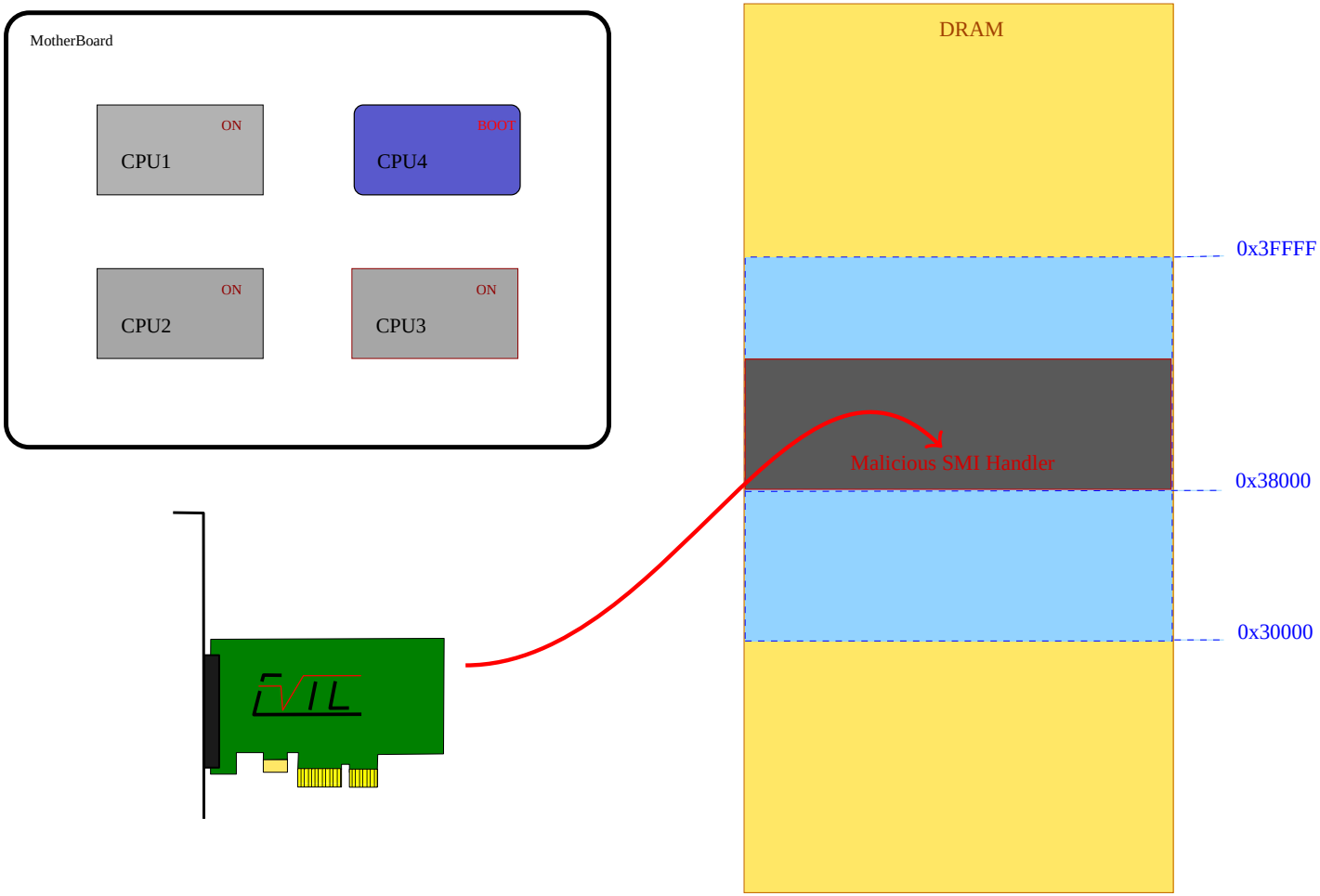
0x38000 attack: Escalate to SMM privileges in a Server



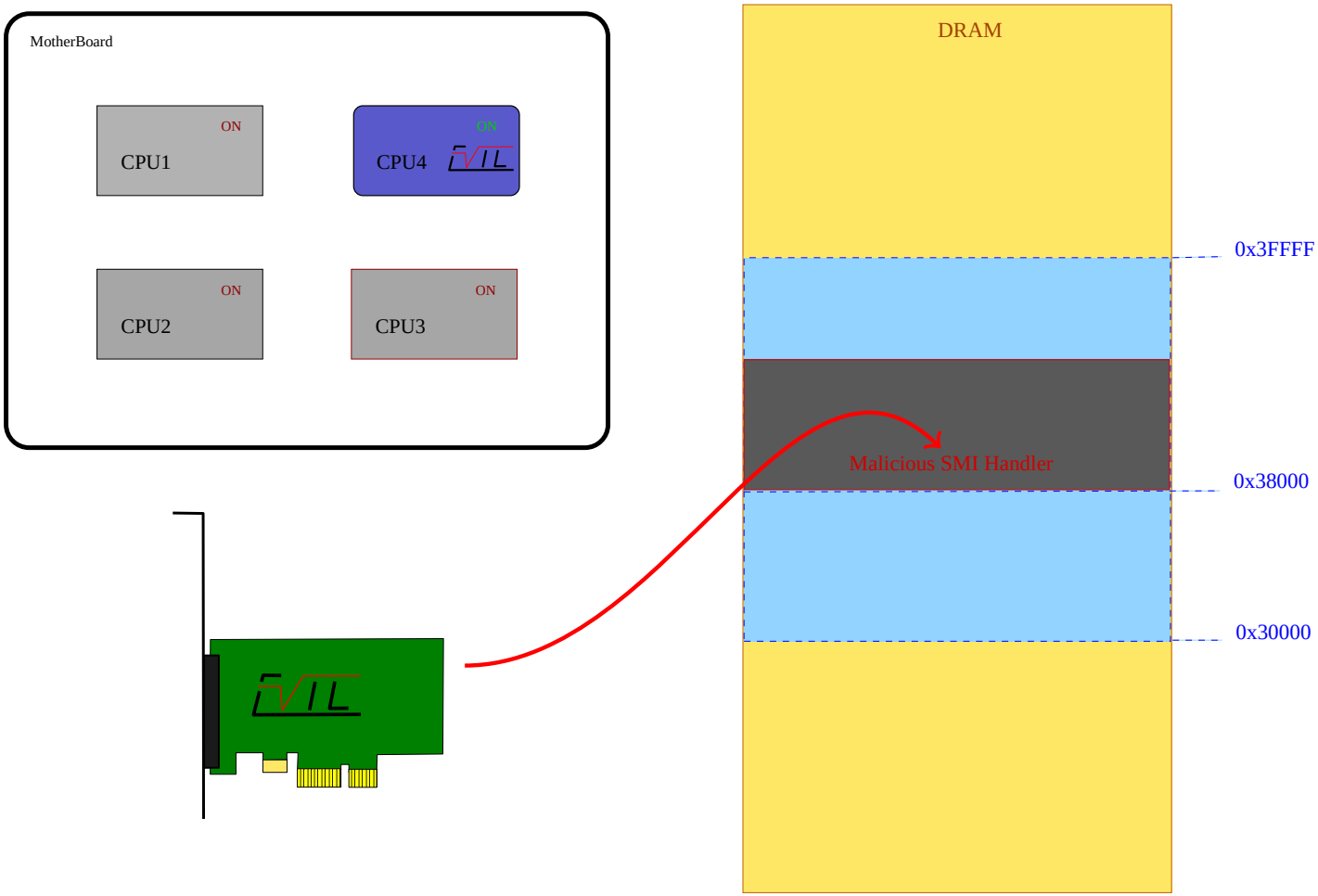




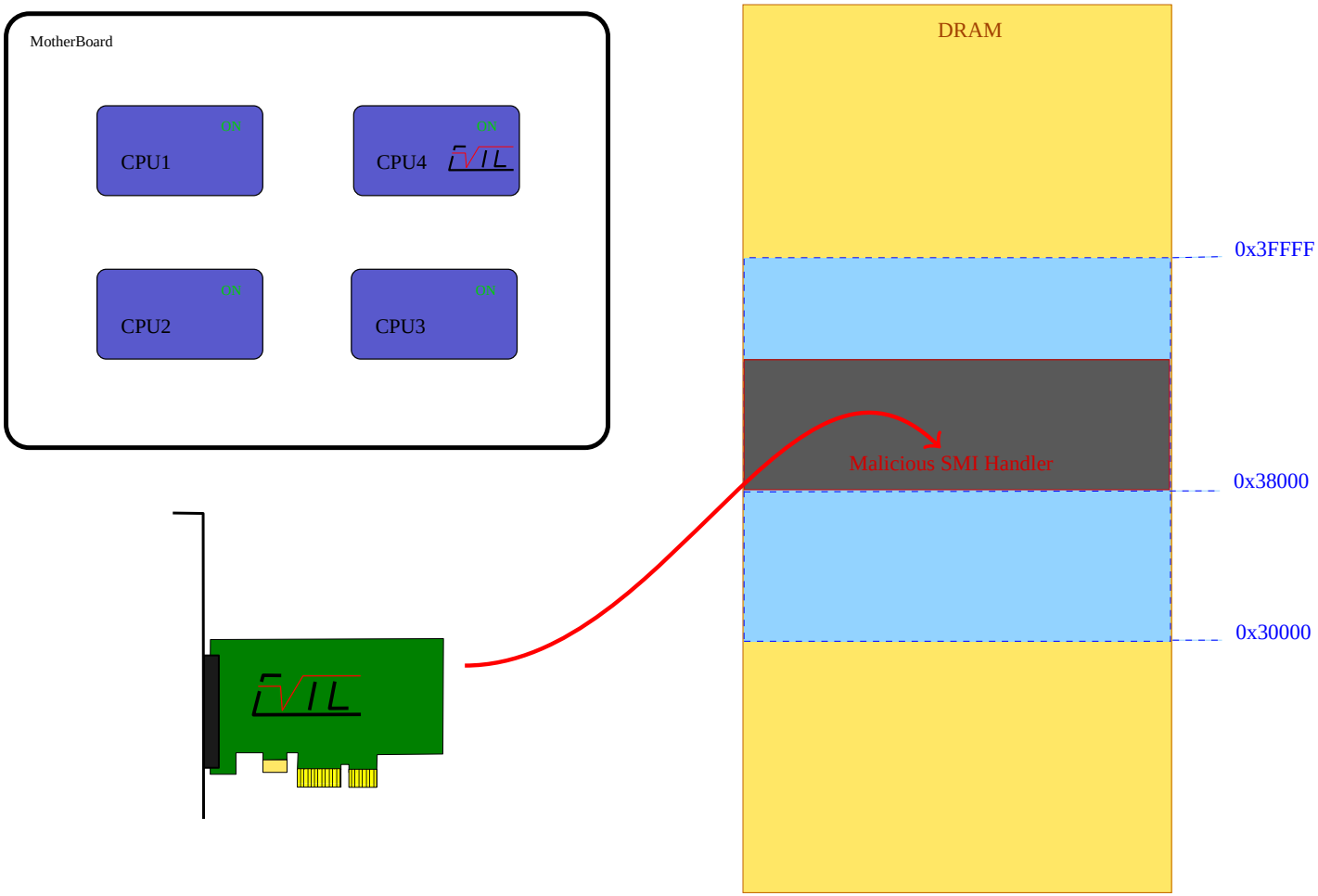
0x38000 attack: Escalate to SMM privileges in a Server



0x38000 attack: Escalate to SMM privileges in a Server



0x38000 attack: Escalate to SMM privileges in a Server



0x38000 attack: Lab Setup

- Hardware
  - Intel Platform/Motherboard supporting CPU Hot-Add
  - Intel Xeon E7-8800 V2 family processor
  - PP3380-AB PCIe 2 x1 - USB3380-AB Evaluation board
  - Attacker's laptop with Windows 10 64bit Operating System

0x38000 attack: Lab Setup

- Hardware
  - Intel Platform/Motherboard supporting CPU Hot-Add
  - Intel Xeon E7-8800 V2 family processor
  - PP3380-AB PCIe 2 x1 - USB3380-AB Evaluation board
  - Attacker's laptop with Windows 10 64bit Operating System
- Firmware
  - BMC Firwmare supporting CPU Hot-Add flow
  - System FW (aka BIOS) supporting CPU Hot-Add Flow

0x38000 attack: Lab Setup

- Hardware
  - Intel Platform/Motherboard supporting CPU Hot-Add
  - Intel Xeon E7-8800 V2 family processor
  - PP3380-AB PCIe 2 x1 - USB3380-AB Evaluation board
  - Attacker's laptop with Windows 10 64bit Operating System
- Firmware
  - BMC Firwmare supporting CPU Hot-Add flow
  - System FW (aka BIOS) supporting CPU Hot-Add Flow
- Software
  - PCILeech solution and a batch script to automate data writes to memory
  - Operating System supporting CPU Hot-Add (i.e. Windows 2008/2012 Server)

PCILeech Configuration

- Place jumper on J3 at PP3380-AB board and start platform
- Run PCILeechFlash_Installer.exe
- Wait a while (1 min or so)
- Shutdown the platform
- Remove jumper on J3 at PP3380-AB board and start platform
- Use our simple batch script to write 0x38000 region to inject arbitrary code

Time to watch the DEMO

Mitigating 0x38000 attack

- The attack just described is possible because DMA engines were still able to inject malicious code in 0x38000 region (despite Hardware effectively prevents code injection from existing cores in the system)

Mitigating 0x38000 attack

- The attack just described is possible because DMA engines were still able to inject malicious code in 0x38000 region (despite Hardware effectively prevents code injection from existing cores in the system)
- To mitigate this, BIOS leverages existing HW protection mechanism in Intel CPUs against rogue DMA engines: GENPROTRANGE register programming

Mitigating 0x38000 attack

- The attack just described is possible because DMA engines were still able to inject malicious code in 0x38000 region (despite Hardware effectively prevents code injection from existing cores in the system)
- To mitigate this, BIOS leverages existing HW protection mechanism in Intel CPUs against rogue DMA engines: GENPROTRANGE register programming
- This mitigation is already in place as part of BIOS reference code delivered to OEMs

0xe2000 attack part I: Take control of the system by inserting rogue code

Corruption of SIPI initialization vector

- DMA malicious writes could be attempted to attack 0xe2000 if not properly protected

0xe2000 attack part I: Take control of the system by inserting rogue code

Corruption of SIPI initialization vector

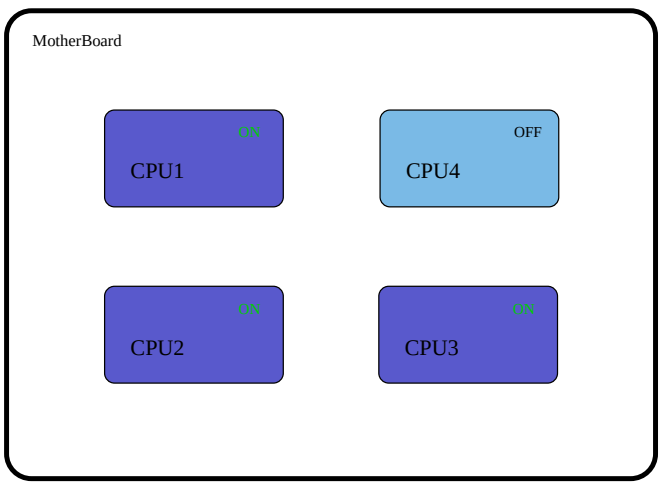
- DMA malicious writes could be attempted to attack 0xe2000 if not properly protected
- However, rogue code already present in other CPUs could try to corrupt the vector either before CPU on-lining flow gets triggered or in between SMIs

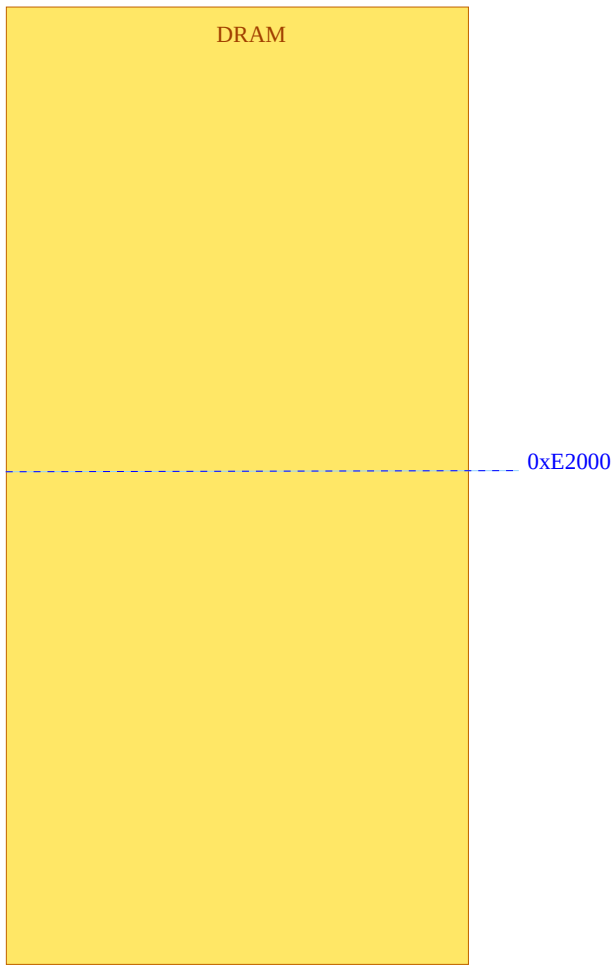
0xe2000 attack part I: Take control of the system by inserting rogue code

Corruption of SIPI initialization vector

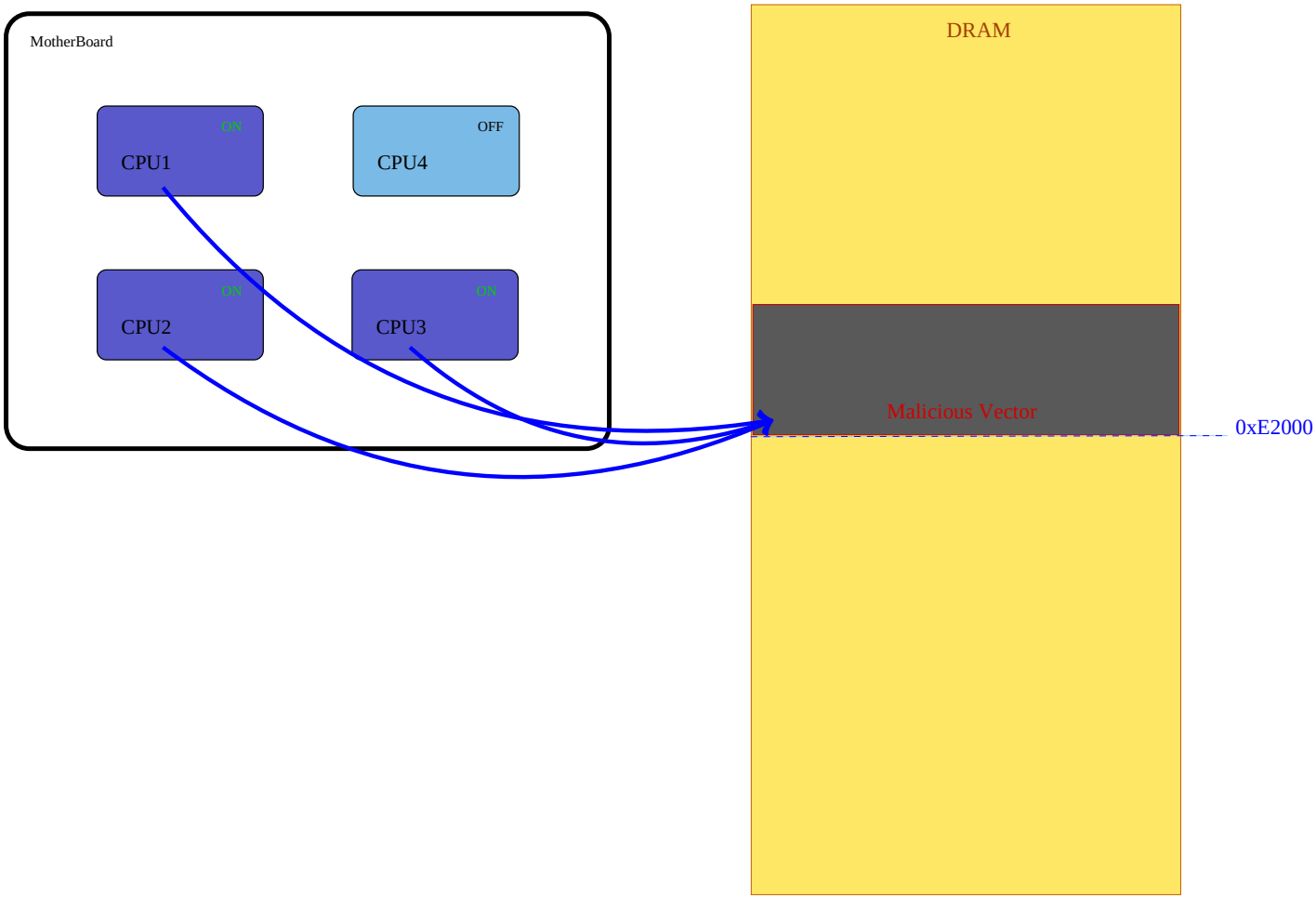
- DMA malicious writes could be attempted to attack 0xe2000 if not properly protected
- However, rogue code already present in other CPUs could try to corrupt the vector either before CPU on-lining flow gets triggered or in between SMIs
- Mitigation: Secure Integrity check before attempting SIPI vector code execution

Depiction of 0xe2000 attack part I

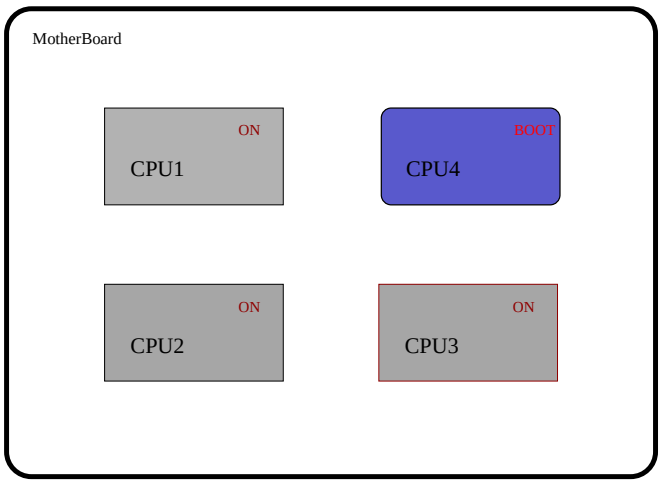


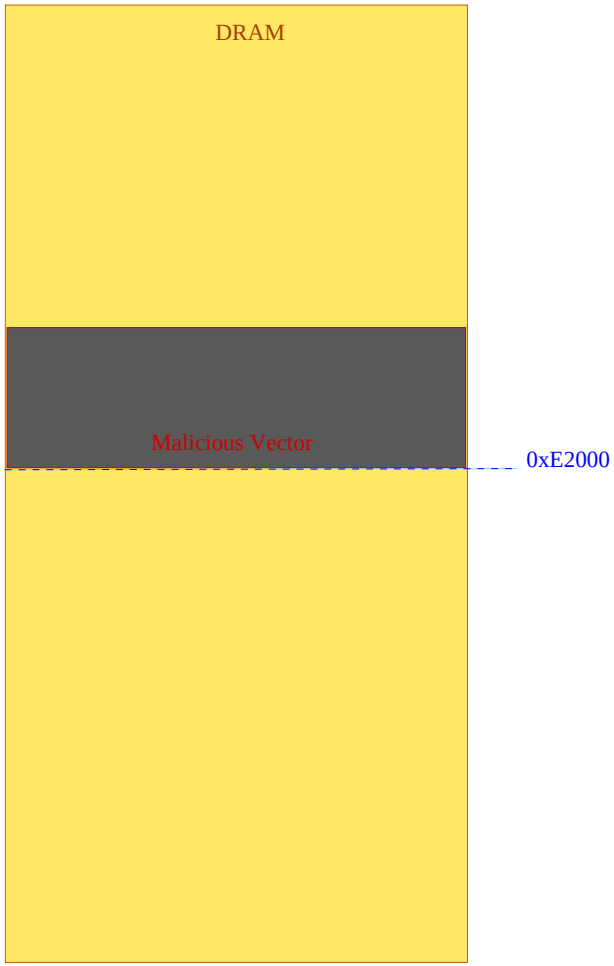


Depiction of 0xe2000 attack part I

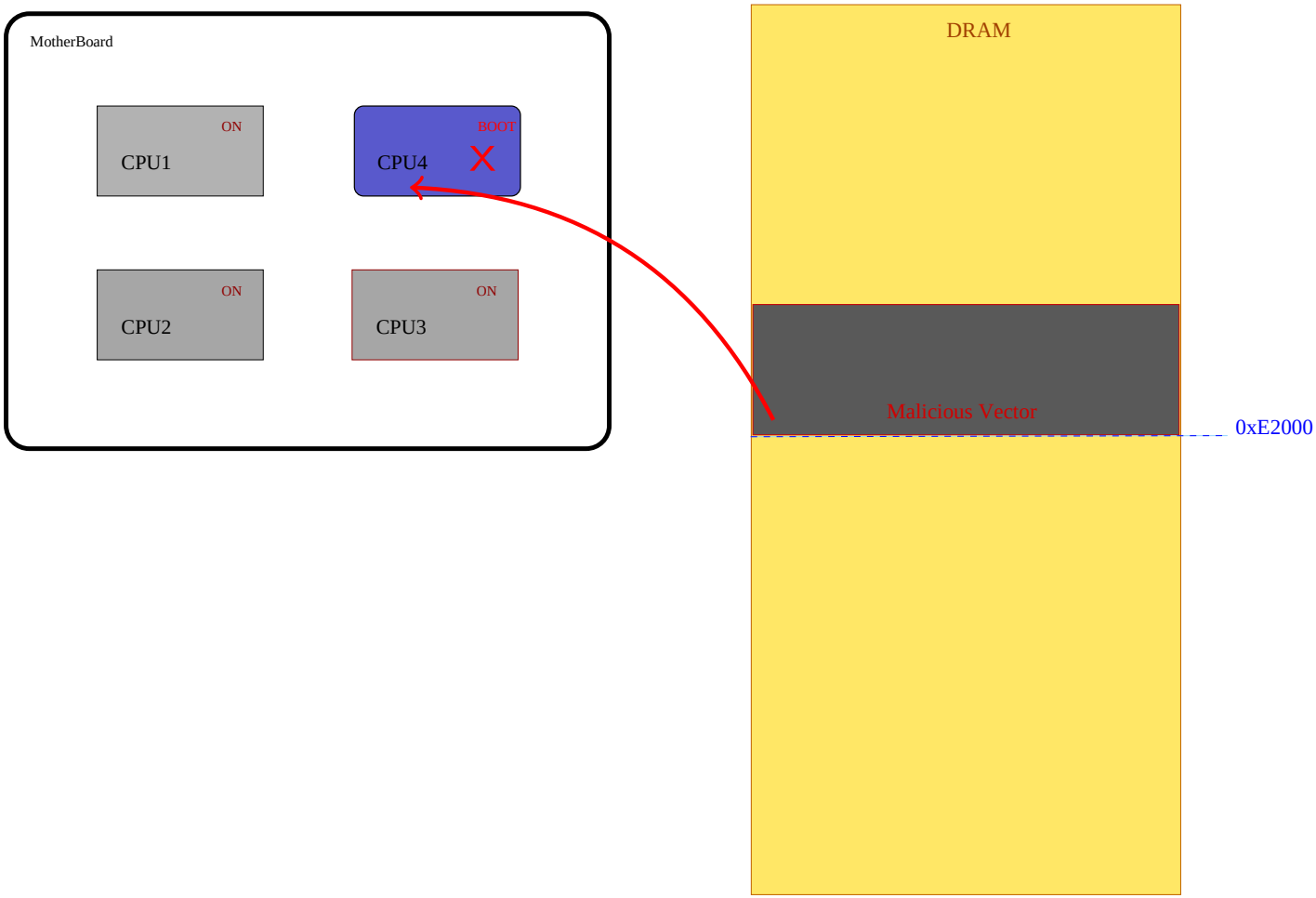


Depiction of 0xe2000 attack part I

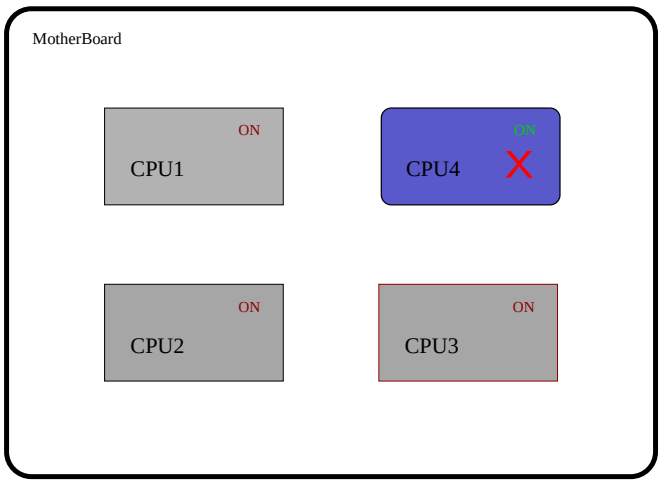


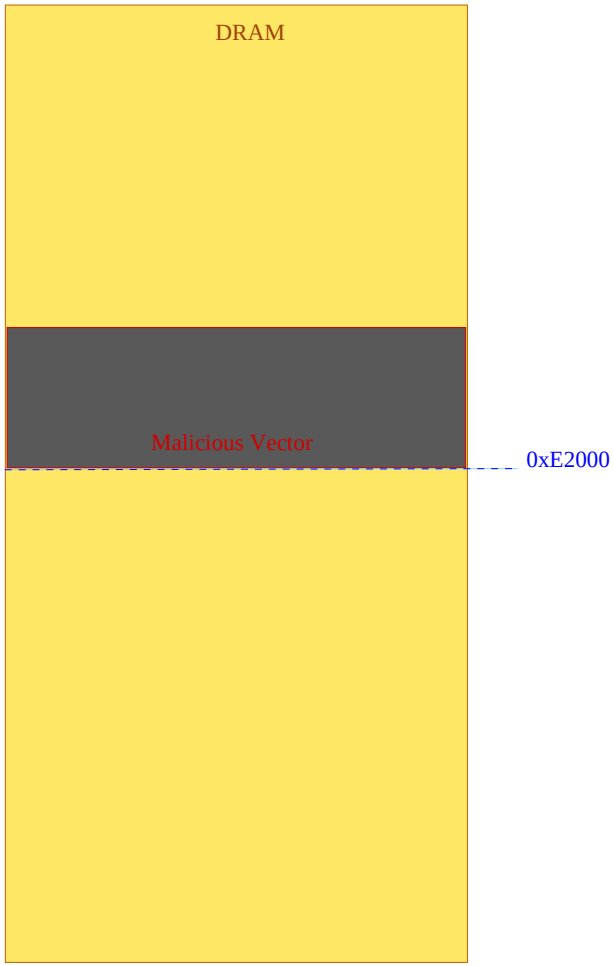


Depiction of 0xe2000 attack part I

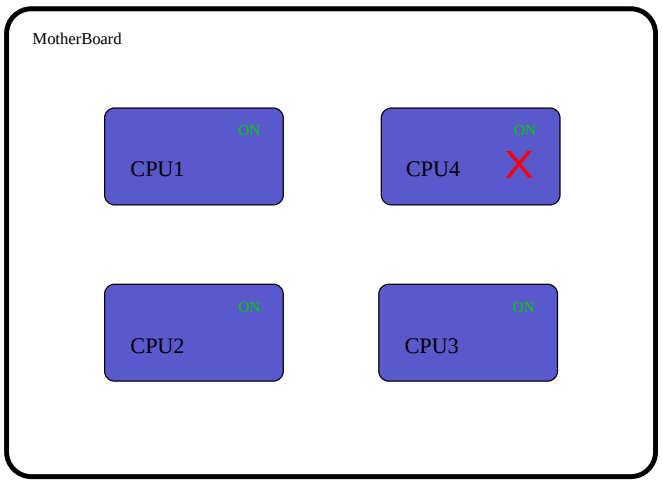


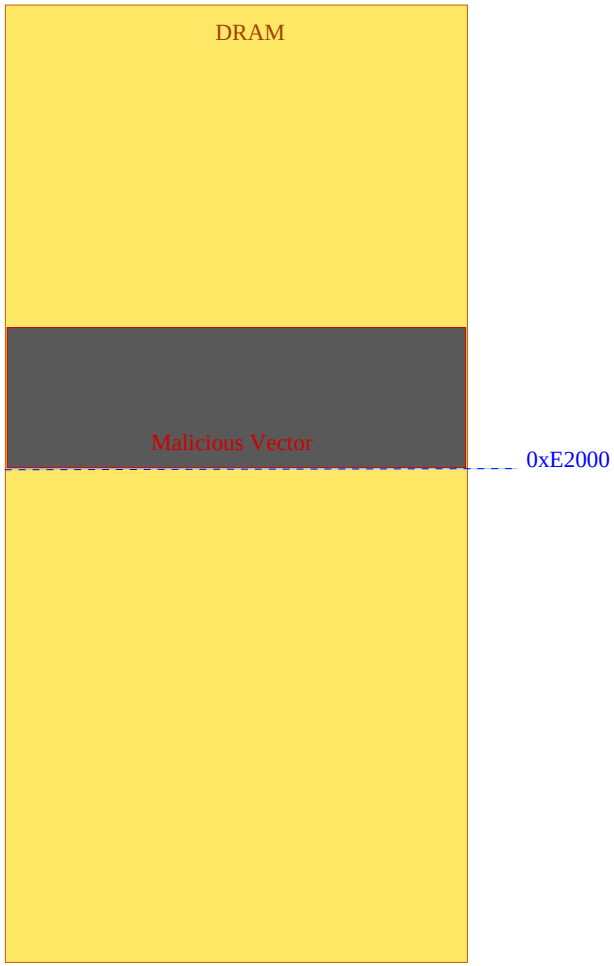
Depiction of 0xe2000 attack part I





Depiction of 0xe2000 attack part I





0xe2000 attack part II: Confusion due to name collision

Integrity verification of 0xe2000 region was meant to be achieved through a cryptographically strong hash function

0xe2000 attack part II: Confusion due to name collision

Integrity verification of 0xe2000 region was meant to be achieved through a cryptographically strong hash function

- It turns out sometimes one can refer to the output of a cryptographic hash function as a *checksum*

0xe2000 attack part II: Confusion due to name collision

Integrity verification of 0xe2000 region was meant to be achieved through a cryptographically strong hash function

- It turns out sometimes one can refer to the output of a cryptographic hash function as a *checksum*
- In fact, this confusion led to erroneously implement an integrity verification mechanism in the form of a weak checksum (from a security standpoint)

0xe2000 attack part II: Confusion due to name collision

Integrity verification of 0xe2000 region was meant to be achieved through a cryptographically strong hash function

- It turns out sometimes one can refer to the output of a cryptographic hash function as a *checksum*
- In fact, this confusion led to erroneously implement an integrity verification mechanism in the form of a weak checksum (from a security standpoint)
- Such mechanism can easily be bypassed by crafting a special rogue code through some tweaks to arithmetically map it to the expected checksum

Mitigating 0xe2000 attack

- Instead of verifying code vector's integrity, always shadow a fresh copy into 0xe2000 region before its execution
- This mitigation is already in place as well by Intel into the BIOS reference code

Conclusions

- Datacenter products and their features deserve a thorough security analysis despite old assumptions of being isolated behind building walls

Conclusions

- Datacenter products and their features deserve a thorough security analysis despite old assumptions of being isolated behind building walls
- DMA remains as an interesting entry point since it might enable remote exploitation of security weaknesses; also, it turns out this entry point might be overlooked while architecturing new technologies

Conclusions

- Datacenter products and their features deserve a thorough security analysis despite old assumptions of being isolated behind building walls
- DMA remains as an interesting entry point since it might enable remote exploitation of security weaknesses; also, it turns out this entry point might be overlooked while architecturing new technologies
- Implementation-wise, it is critical to ensure developers correctly understand the exact security mechanism that mitigates a corresponding threat; failure in this regard could lead to mistakenly break overall system's security

Thank you!

Time for Q&As