

# I/O Attacks in Intel-PC Architectures and Countermeasures

Fernand Lone Sang, Vincent Nicomette and Yves Deswarte

Laboratoire d'Analyse et d'Architecture des Systèmes  
LAAS-CNRS – Toulouse, France

July 6, 2011



# Context and issues

Protecting information systems is difficult:

- complexity of such systems keeps on increasing
- attack surface on such systems keeps on expanding

Main attack vectors on an information system:

- 1 execution of some malicious code (malware) by the processor
  - exploitation of a vulnerability  
→ buffer overflow, format strings, ...
  - system features abuse  
→ kernel modules, virtual devices, ...
- 2 misuse of Input/Output mechanisms
  - Direct Memory Access (DMA)
  - interrupt mechanism
  - other I/O mechanisms

# Context and issues

Protecting information systems is difficult:

- complexity of such systems keeps on increasing
- attack surface on such systems keeps on expanding

Main attack vectors on an information system:

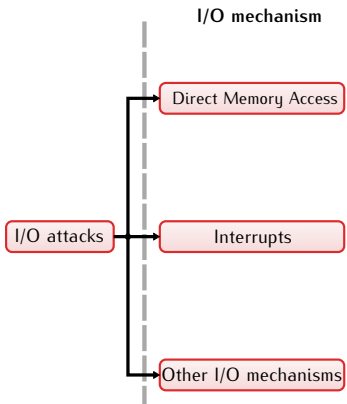
- 1 execution of some malicious code (malware) by the processor
  - exploitation of a vulnerability  
→ buffer overflow, format strings, ...
  - system features abuse  
→ kernel modules, virtual devices, ...
- 2 misuse of Input/Output mechanisms
  - Direct Memory Access (DMA)
  - interrupt mechanism
  - other I/O mechanisms

# I/O attack vectors

To perform an I/O attack, an attacker can:

- use a regular I/O controller
  - abuse the I/O controller's control interface
    - needs to execute some malicious code on the processor
    - needs to get I/O privileges to interact with the I/O controller
  - exploit a vulnerability in the I/O controller's firmware
    - does not need to execute some malicious code on the processor
      - malicious code is executed on the I/O controller's embedded processor
    - enables the attacker to define its own control interface
- develop a dedicated I/O controller (e.g., using FPGA)
  - use an attacker-defined control interface
    - provides more flexibility to the attacker
    - developed generally for specific purposes

# I/O-based attacks tree



# Direct Memory Access mechanism

What does Direct Memory Access mechanism stand for ?

- I/O mechanism that enables an I/O controller
  - to perform directly a data transfer to/from the main memory
  - to offload the CPU of these transfers
- relies on a dedicated DMA engine

# Direct Memory Access mechanism

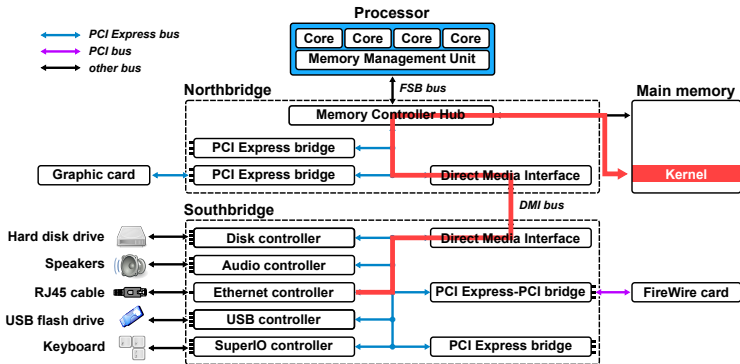
What does Direct Memory Access mechanism stand for ?

- I/O mechanism that enables an I/O controller
  - to perform directly a data transfer to/from the main memory
  - to offload the CPU of these transfers
- relies on a dedicated DMA engine

Examples of I/O controllers using DMA:

- network controllers (WiFi, Ethernet, ...)
  - *e.g.*, to transfer network frames into/from the main memory
- disk controllers
  - *e.g.*, to transfer files into/from the main memory
- graphic controllers
  - *e.g.*, to transfer textures, buffer objects from the main memory

# DMA-based attacks (1/2)

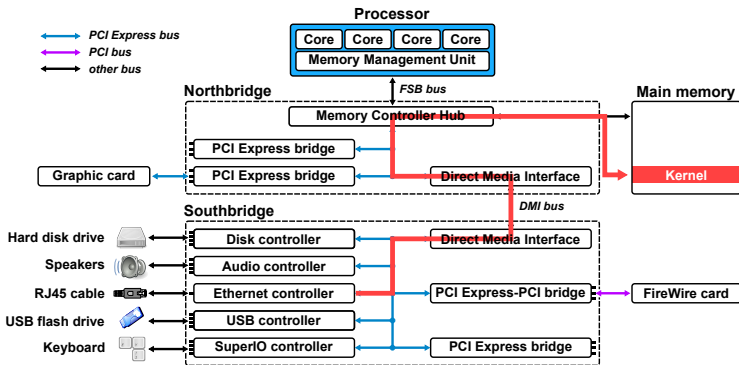


DMA attacks aiming at the main memory:

- ☺ attack (confidentiality & integrity) on software components



# DMA-based attacks (1/2)

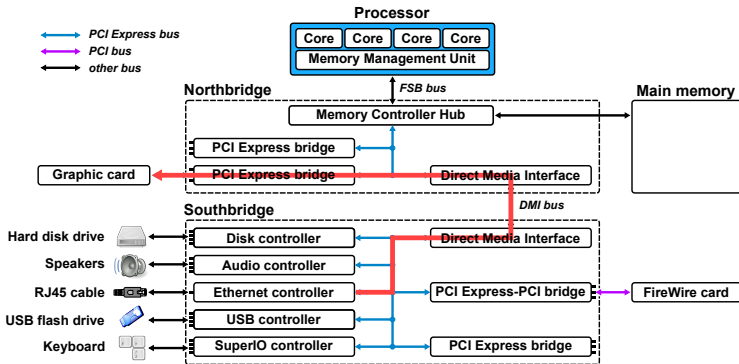


DMA attacks aiming at the main memory:

- ☺ attack (confidentiality & integrity) on software components
- ☹ modifications made in the main memory can be detected

**Examples:** [Dornseif 04, Becher 05, Carrier 04, Nick L. Petroni 04, Maynor 05, Boileau 06, Duflot 07, Duflot 10, Aumaitre 09, Piegdon 07]

## DMA-based attacks (2/2)

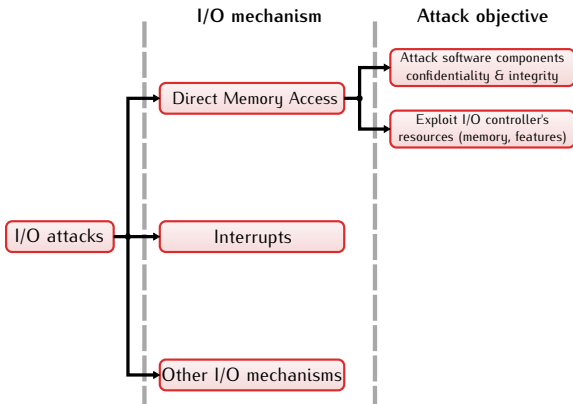


DMA attacks aiming at I/O controllers' internal memory:

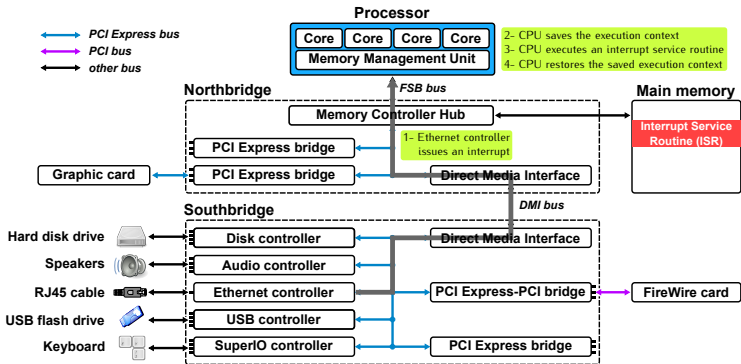
- ☺ exploit I/O controllers' resources (memory, features, ...)
- ☺ no modifications in the main memory, hard to detect

Examples: [Dornseif 04, Triulzi 08, Triulzi 10, Lone Sang 11a]

# I/O-based attacks



# Interrupt mechanism



What does the interrupt mechanism stand for ?

- enables a controller to signal the CPU a need for attention
- enables the CPU to avoid wasting cycles to perform polling loops

# Interrupt-based attacks

What can attacker do with an interrupt ?

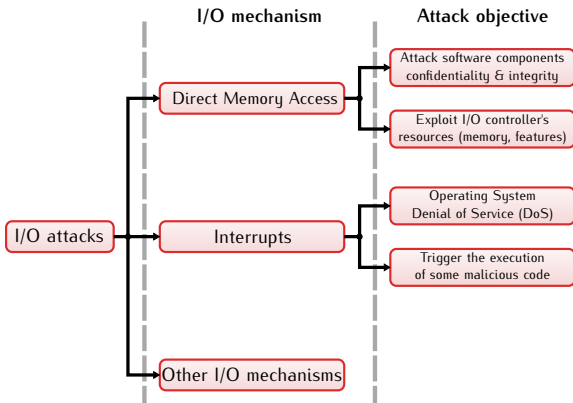
- denial of service
  - attacker makes I/O controllers generate an interrupt storm
  - OS kernel will waste CPU cycles to handle interrupts

**Example:** [Liguori 09]

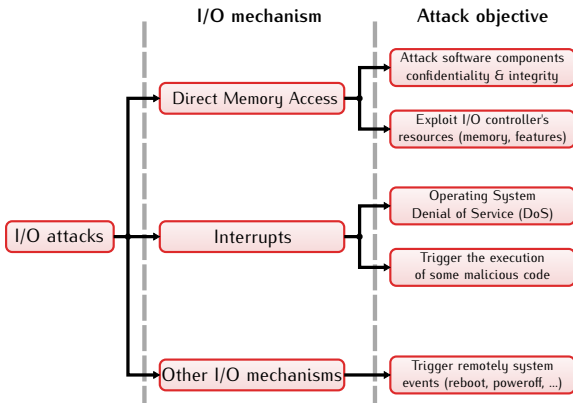
- trigger the execution of some malicious code
  - attacker hides some malicious code at the address of an ISR
    - with the cooperation of the processor
    - using DMA attacks
  - attacker makes an I/O controller generate an interrupt

**Example:** [Wojtczuk 11]

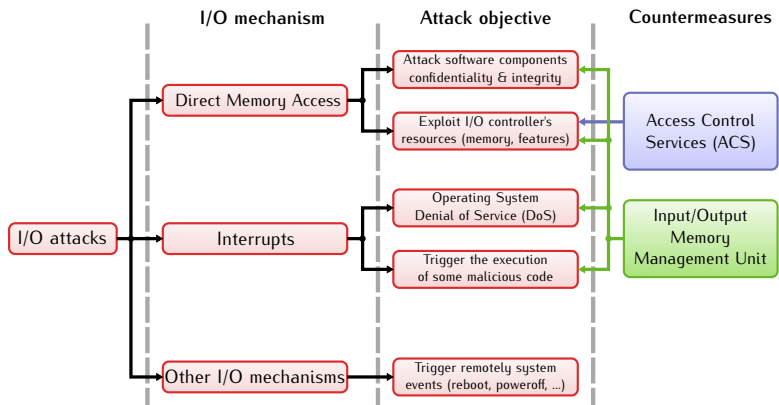
# I/O-based attacks tree



# I/O-based attacks tree



# I/O-based attacks countermeasures





# Input/Output Memory Management Unit (1/2)

What is an I/O Memory Management Unit (I/O MMU) ?

- component similar to the Memory Management Unit in the CPU
  - virtualizes the main memory
  - filters devices' access to it
  - configured through page tables stored in the main memory
- memory management unit dedicated to I/O controllers

# Input/Output Memory Management Unit (1/2)

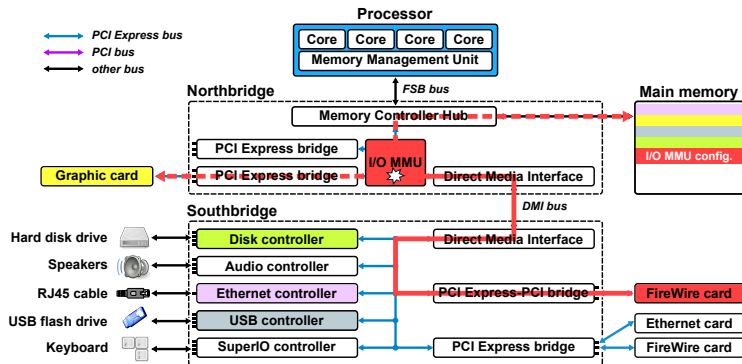
What is an I/O Memory Management Unit (I/O MMU) ?

- component similar to the Memory Management Unit in the CPU
  - virtualizes the main memory
  - filters devices' access to it
  - configured through page tables stored in the main memory
- memory management unit dedicated to I/O controllers

How can an I/O MMU enhance platform security ?

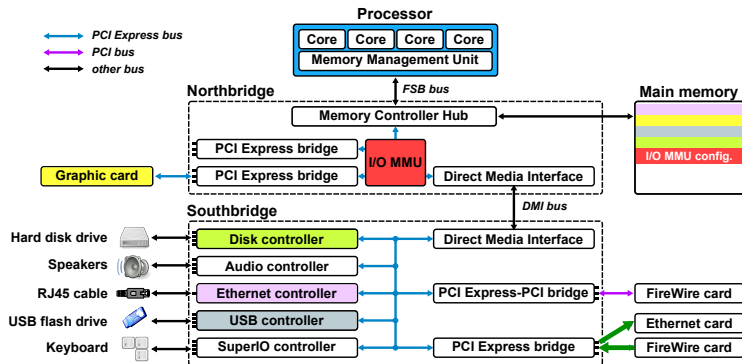
- ensures isolation between I/O controllers' memory regions
  - associates a domain and some memory regions to an I/O controller
  - restricts I/O controllers' accesses only to their respective domains
- remaps and filters interrupts

# Input/Output Memory Management Unit (2/2)



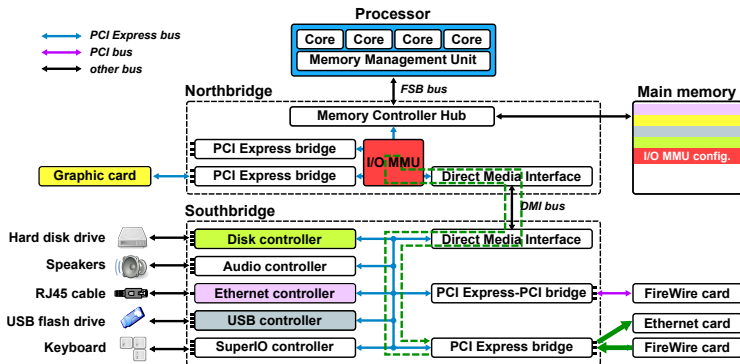
☺ I/O MMU controls efficiently access to the northbridge

# Input/Output Memory Management Unit (2/2)



- ☺ I/O MMU controls efficiently access to the northbridge
- ☹ I/O MMU has some limitations:
  - I/O controller ID spoofing/sharing [Lone Sang 10]
  - DMA peer-to-peer attacks [Lone Sang 11b]
  - I/O MMU bypass through interrupts [Wojtczuk 11]

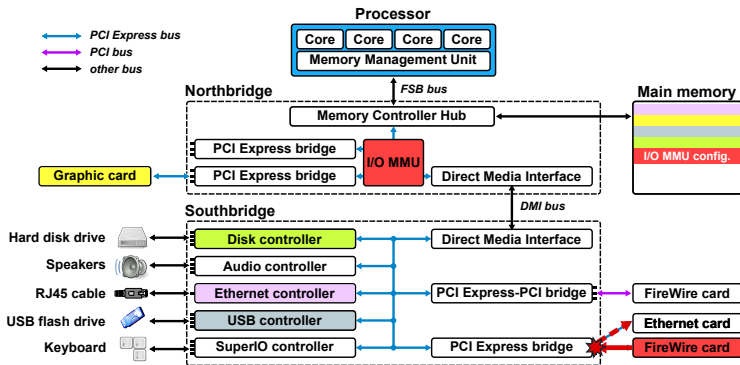
# Access Control Services (1/2)



How can Access Control Services (ACS) enhance security ?

- enable the OS to configure I/O bridges to perform access control
  - ACS Upstream Forwarding (U)

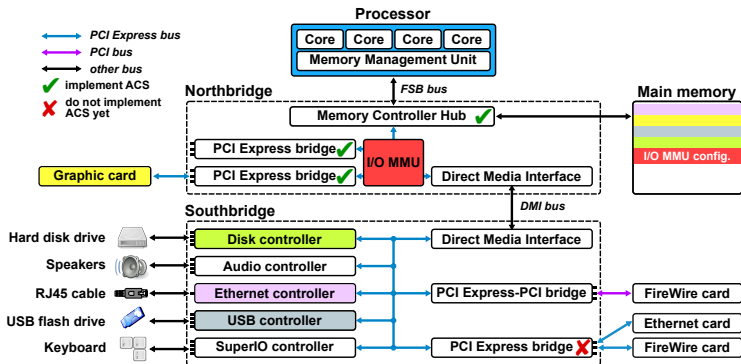
# Access Control Services (1/2)



How can Access Control Services (ACS) enhance security ?

- enable the OS to configure I/O bridges to perform access control
  - ACS Upstream Forwarding (U)
  - ACS P2P Egress Port (E)
  - ...

## Access Control Services (2/2)



ACS extensions in current chipsets:

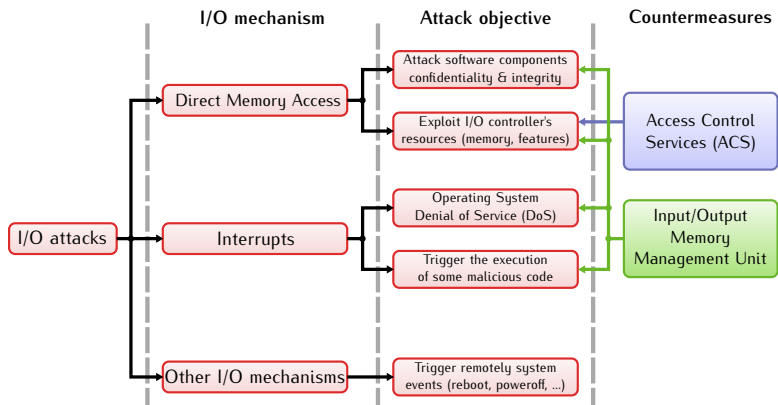
- recently implemented in chipsets, precisely in the northbridge
- by default, not activated and has to be configured manually

## Conclusion (1/2)

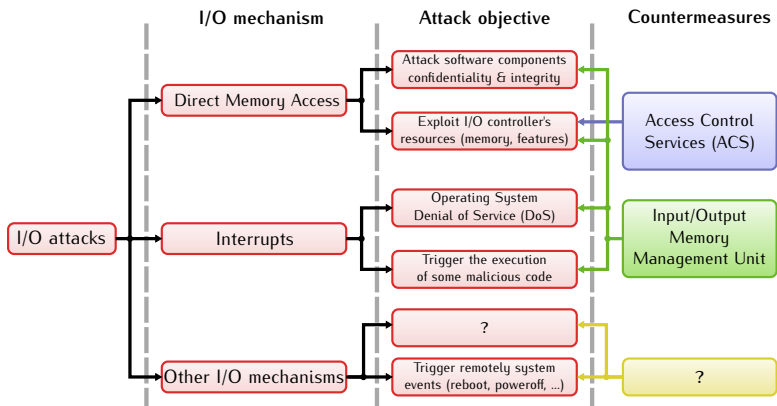
	Use a regular I/O controller		Develop a custom I/O controller
	Misuse of an I/O controller's programming interface	Exploitation of a vulnerability in an I/O controller's firmware	Using an attacker-defined programming interface
Physical access and code execution needed	[Dufлот 07, Gazet 11, Wojtczuk 11]	[Boileau 06, Dufлот 10]	[Devine 09, Aumaitre 10]
Physical access without code execution needed	[Dornseif 04, Becher 05, Maynor 05, Boileau 06, Aumaitre 09, Piegdon 07]	-	[Carrier 04, Nick L. Petroni 04]
Remote access	-	[Dufлот 10, Delugré 10]	No examples yet to our knowledge



## Conclusion (2/2)



# Future work



Thank you for your attention ...

**Any questions ?**

# Bibliography 1



Damien Aumaitre.

*A Little Journey Inside Windows Memory.*  
volume 5, pages 105–117. Springer, 2009.



Damien Aumaitre & Christophe Devine.

*Subverting Windows 7 x64 Kernel with DMA attacks.*  
In HITBSecConf 2010 Amsterdam, 29 June - 2 July 2010.



Michael Becher, Maximillian Dornseif & Christian N. Klein.

*FireWire - all your memory are belong to us.*  
In CanSecWest/core05, 4–5 May 2005.



Adam Boileau.

*Hit by a Bus: Physical Access Attacks with FireWire.*  
In RUXCON 2006, October 2006.

## Bibliography 2



Brian Carrier & Joe Grand.

*A Hardware-based Memory Acquisition Procedure for Digital Investigations.*

Digital Investigation, vol. 1, no. 1, pages 50–60, February 2004.



Guillaume Delugré.

*Closer to metal: reverse-engineering the Broadcom NetExtreme's firmware.*

In Hack.lu, Luxembourg, 27–29, October 2010.



Christophe Devine & Guillaume Vissian.

*Compromission physique par le bus PCI.*

In Proceedings of the 7th Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC 2009), pages 169–193, June 2009.



Maximillian Dornseif.

*Owned by an iPod - Hacking by Firewire.*

In PacSec/core04, 11–12 November 2004.

## Bibliography 3



Loïc Duflot.

*Contribution à la sécurité des systèmes d'exploitation et des microprocesseurs.*

PhD thesis, Université de Paris XI, October 2007.



Loïc Duflot, Yves-Alexis Perez, Guillaume Valadon & Olivier Levillain.

*Can you still trust your Network Card?*

In CanSecWest/core10, 24–26 March 2010.



Alexandre Gazet.

*Sticky fingers & KBC Custom Shop.*

In Proceedings of the 9th Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC 2011), pages 175–189, June 2011.



Anthony Liguori.

*Re: A few KVM security questions*, 7December 2009.

## Bibliography 4



Fernand Lone Sang, Eric Lacombe, Vincent Nicomette & Yves Deswarte.

*Exploiting an I/O MMU Vulnerability.*

In Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software (MALWARE), pages 7–14, 19–20 October 2010.



Fernand Lone Sang, Vincent Nicomette & Yves Deswarte.

*Demonstration of a peer-to-peer DMA Attack against the Framebuffer of a Graphic Controller Through FireWire*, January 2011.



Fernand Lone Sang, Vincent Nicomette, Yves Deswarte & Loïc Dufлот.

*Attaques DMA peer-to-peer et contremesures.*

In Proceedings of the 9th Symposium sur la Sécurité des Technologies de l'Information et des Communications (SSTIC 2011), June 2011.

(to be published soon at: <http://www.sstic.org/2011/actes/>).

## Bibliography 5



David Maynor.

*Own3d by everything else - USB/PCMCIA Issues.*  
In CanSecWest/core05, 4-5 May 2005.



Jr. Nick L. Petroni, Timothy Fraser, Jesus Molina & William A. Arbaugh.

*Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor.*  
In 13th USENIX Security Symposium, 9-13 August 2004.



David R. Piegdon.

*Hacking in Physically Addressable Memory.*  
In Seminar of Advanced Exploitation Techniques, WS 2006/2007,  
12 April 2007.



Arrigo Triulzi.

*Project Moux Mk.II - "I Own the NIC, Now I want a Shell!"*  
In PacSec/core08, 12-13 November 2008.



## Bibliography 6



Arrigo Triulzi.

*The Jedi Packet Trick takes over the Deathstar (or: "Taking NIC Backdoors to the Next Level").*

In [CanSecWest/core10](#), 24-26 March 2010.



Rafal Wojtczuk & Joanna Rutkowska.

*Following the White Rabbit: Software Attacks against Intel VT-d.*

Rapport technique, [Invisible Things Lab \(ITL\)](#), May 2011.