Christopher Bero
CPE 436
Exam 2

What increases in security does an IOMMU provide?

One of the longstanding cruxes of computer performance is the relationship between compute and storage. The effect is so great that the two leading schools of design in computer architecture – von Neumann and Harvard – differ most noticeably in regard to their storage access methods. Von Neumann systems, the majority of modern computers, have a single bus and address space for processor instructions and cache memory operations. In contrast, Harvard systems employ a separate set of dedicated buses and address space for memory operations, and a separate set for instructions. The task of delivering greater memory operation performance has undergone intense investigation and has resulted in a complex and carefully tuned paging and caching system to make the best possible use of mixed fast and slow memory. Another method used to handle larger blocks of memory without loading the main processor unfairly with move operations is direct memory access (DMA). DMA allows the central processor to instigate a move operation for a hardware device, continue executing other instructions, and then return when the DMA controller is finished. Processors can have integrated memory management units (MMU) which divide a virtual address space up into pages which correspond to physical memory addresses. These virtual address spaces are then assigned to individual programs, which execute on it similarly to direct physical memory access. The MMU helps avoid memory fragmentation, where the physical address space becomes discontinuous once memory has been freed by previous processes; with virtual memory, the smaller sections of free space be mapped to a contiguous block of virtual addresses. MMUs also offer memory protection, where a process is unable to errantly access memory addresses of other processes.

But what do we do about DMA capable hardware devices? These IO buses can interact with the main memory without the same level of protection provided by an MMU to processes. This is where the IO memory management unit (IOMMU) comes into play. The IOMMU does exactly what we would expect given the background on MMUs: it is an MMU which mediates between a DMA IO bus for a hardware component and the main memory. Like normal MMUs, the IOMMU translates virtual address spaces to physical addresses and can offer similar memory protection schemes as well. When using an IOMMU, fragmented memory can be allocated to contiguous virtual addresses just like an MMU. IOMMUs also play a heavy role in virtualization; they can map physical memory that normally falls outside 32 bit address spaces down to allow a 32 bit guest operating system access memory normally unavailable to it, and can mediate between a guest operating system's memory space and a DMA to the host system's memory.

To see how an IOMMU also aids in security, let's consider the following hypothetical situation. A defective PCIe device will operate normally for roughly 18 hours (65,535 seconds) at which point its 16 bit internal clock overflows and corrupts the FPGA's main memory address for a struct. The next time this PCIe device writes

across the virtual bus to the system's main memory, it will be access the wrong region of main memory. The PCIe device has just overwritten memory owned by Microsoft Word, and when Word crashes due to the erroneous values a term paper is lost! But what if the system included an IOMMU? Then the poorly designed PCIe device would only have access to a virtual address space as mediated by the IOMMU. When the bad memory addresses are written, they are translated to physical addresses by the IOMMU, and the bad values are written out to a section of physical memory that is now owned by the PCIe device. The term paper is now safe!

This kind of memory protection has received significantly more attention in recent years, long after the performance considerations mentioned at the beginning of this section. As a result, proper implementation of AMD/Intel IOMMU devices in the Linux kernel has lagged behind the need, and allows broken or malicious devices too much access. As we discussed in class, hopefully this IOMMU support will arrive in stable kernels soon, such that end users can enjoy the security benefit.