Border Control: Sandboxing Accelerators

Lena E. Olson Jason Power Mark D. Hill David A. Wood
University of Wisconsin-Madison
Department of Computer Sciences

1. Summary of paper

In the era of dark silicon, hardware accelerators are one way to improve performance and reduce power, and therefore they have been proposed for a variety of applications. Some accelerators may be custom-designed by third parties, and purchased as soft or firm intellectual property (IP). At the same time, groups like the HSA Foundation are pushing for greater programmability and tighter integration between CPUs and accelerators, through interfaces such as shared virtual memory.

Although allowing accelerators access to system memory makes sense from a programmability and performance standpoint, it has security implications that have yet to be fully explored. In particular, third-party accelerators may have bugs or design flaws, or they may even be malicious. Incorrect accelerator accesses to host memory can have serious consequences, because they allow the accelerator to interfere with processes which never run on the accelerator. These consequences include corruption of operating system structures and leakage of information between processes.

The *Principle of Least Privilege* states that "Every program and every user of the system should operate using the least set of privileges necessary to complete its job. Primarily, this principle limits the damage that can result from an accident or error." [9]. Similarly, an accelerator should only be able to access memory addresses associated with the process it is running. This limits the damage it can cause, protecting all processes that do not run on the accelerator from corruption or information leakage.

Our work, *Border Control*, provides sandboxing: isolation and protection between accelerators and the host system. In particular, Border Control prevents untrusted accelerators from making invalid accesses to the host memory system.

1.1. Threat Model

The threat model we address is an untrusted accelerator accessing a host physical memory location in violation of the permissions set by the trusted operating system. We assume that the host operating system and host hardware are trusted, but do not place any constraints on the correctness of accelerators or the software running upon them. The *threat vector* is any untrusted accelerator that has direct access to physical memory. The *addressed threats* are violations of *confidentiality (or integrity)* of host memory, if the untrusted accelerator *reads (or writes)* a host physical address to which it is not currently granted *read (or write)* access.

	Provides Protection		Direct Access
	For OS	Between Processes	to physical memory
ATS-only IOMMU	✗	✗	✓
Full IOMMU	✓	✓	✗
IBM CAPI	✓	✓	✗
ARM TrustZone	✓	✗	✓
Border Control	✓	✓	✓

Table 1: Comparison of Border Control with other approaches.

A confidentiality violation could allow an accelerator to read sensitive data from other processes or from the OS. An integrity violation could allow the accelerator to corrupt critical data. In the presence of a malicious accelerator (or an exploitably buggy accelerator) which can access host memory without restrictions, an attacker can gain full control of the system through reading and writing OS data on the host. This is true even if the malicious accelerator is not meant to handle sensitive or critical data (e.g., a video decoder), so long as it has direct access to host physical memory.

1.2. Previous Approaches

A number of approaches have been used by industry to make accelerators more programmable and safer than devices that use physical addresses (e.g., classic DMA). However, these approaches have focused either on safety or high performance, but not both (see Table 1). For example, approaches such as the I/O Memory Management Unit (IOMMU) [3, 4, 6] and CAPI [5, 10] can provide safety by requiring that *every* memory request be made by virtual address, then translated and checked by a trusted hardware interface. However, they prevent the accelerator from implementing TLBs or coherence-friendly physically addressed caches. These constraints may be acceptable for accelerators with very regular memory access patterns or low performance requirements, but some accelerators require higher performance.

In contrast, other approaches prioritize high performance over safety. Some current high-performance GPU configurations use the IOMMU's Address Translation Service (ATS) for translation, but allow the accelerator to store physical addresses in an accelerator TLB and caches [3, 4, 6]. The accelerator is then able to *bypass* the IOMMU and make requests directly to memory by physical address, without any check of whether the address is legitimate. Some orthogonal techniques,

Border Control: Sandboxing Accelerators, Lena E. Olson, Jason Power, Mark D. Hill, David A. Wood. Published in MICRO-48, December 2015.

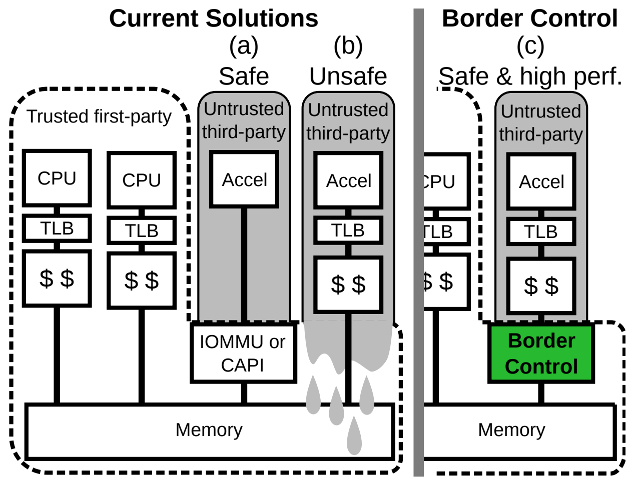

Figure 1: An example system with untrusted third-party accelerators interacting with the system in three ways: a safe but slow IOMMU (a), a fast but unsafe direct access (b), and Border Control, which is safe and high performance (c).

such as ARM TrustZone [1], protect highly sensitive data, but offer no protection between normal processes.

1.3. Border Control

Border Control aims to overcome the limitations of existing approaches by allowing accelerators to use performance optimizations such as TLBs and physical caches, while also guaranteeing that memory access permissions are respected by accelerators, regardless of design errors or malicious intent (see Figure 1). Border Control decouples address translation and permission checking, and thus it does not need to perform reverse address translation. Instead, Border Control checks the physical address of every memory access from the untrusted accelerator to the shared memory on the trusted host and ensures that the process running on the accelerator has permissions for that physical page. We build upon the existing process abstraction and the page table to determine whether an access should be allowed. If an accelerator attempts a bad access, Border Control blocks the access and notifies the operating system.

1.4. Border Control Design

Border Control has two main components: the *Protection Table* and the *Border Control Cache (BCC)*. The Protection Table is a flat table in physical host memory, with a read and a write permission bit for each page of physical memory. It is indexed by physical page number (PPN) and initialized to zero (no permissions).

Every time the trusted ATS performs a translation on behalf of the accelerator, it provides the PPN and associated permission bits to the accelerator TLB. In our approach, it also provides this information to Border Control, which updates the Protection Table accordingly. For an access to host physical memory to be valid, the accelerator must have obtained the PPN from the ATS. Therefore, Border Control allows the access to proceed if and only if the appropriate permission bit in the Protection Table is set. The Protection Table provides a correct implementation and full safety.

Because accessing memory on every access to the host memory has a latency overhead, Border Control also caches recently seen permissions in a small structure: the BCC. Each entry in the BCC can contain permissions for multiple consecutive physical pages. This structure can be very small and still achieve a high hit rate due to spatial locality, because a single 64-byte entry can provide protection information for 256 pages at 2 bits per page. This is much smaller than would be required for an address translation.

Please refer to the full paper for detailed descriptions of the actions Border Control takes on different permission events (e.g., permission downgrades).

1.5. Results

We quantitatively evaluated Border Control for a general purpose GPU (GPGPU). GPGPUs are a prominent high-performance accelerator which is capable of high memory traffic and irregular memory access patterns. Thus, the GPGPU is a stress-test for memory safety mechanisms. We compared performance with Border Control to the unsafe ATS-only IOMMU as a baseline, as well as to configurations similar to the full IOMMU and CAPI. We evaluated performance for a subset of the Rodinia benchmarks, for both a highly-threaded GPGPU (high-performance accelerator proxy) and a moderately-threaded GPGPU (more latency-sensitive accelerator proxy).

Our results show that current protection mechanisms degrade performance significantly for high-performance accelerators. Compared to the unsafe case, the full IOMMU had an average slowdown of 374% and 85% for a highly-threaded and moderately-threaded GPGPU, respectively. The CAPI-like configuration had an overhead of 3.81% and 16.5%, respectively.

Border Control provides the same safety as the full IOMMU and CAPI with almost no performance degradation. Border Control shows an average of 0.15% (highly-threaded) and 0.48% (moderately-threaded) slowdown versus the unsafe baseline. Our results show that Border Control can guarantee safety for the host system without sacrificing the high performance of the ATS-only IOMMU baseline.

Additionally, Border Control has low space overhead. The Protection Table contains two bits per page of physical memory, or 0.006% of memory capacity for each active accelerator. For the BCC, we found that an approximately 1KB cache (including tags and data) was sufficient for our workloads, due to the large reach of each entry.

These results show that Border Control is an effective means of providing safety while still retaining high performance.

2. Case for influence

This paper introduces and addresses a novel security problem in the emerging third-party accelerator economy. Even today, consumer-facing corporations incorporate IP designed elsewhere into their devices. For instance, consumers buy iPhones from Apple, and the SoC which powers the phone is engineered by Apple. However, many of the components on the SoC were developed by third-party IP providers, like the GPU IP from Imagination Technologies.

It it hypothesized that there will be an increasing number of third-party accelerators [7]. Efforts like open source hardware [2] will lower the barrier to entry for new IP providers.

With the proliferation of accelerators, it will be increasingly important for consumer-facing businesses to have a way to protect their systems from both hardware bugs and malicious designs. If an accelerator is buggy or malicious, it may be a vector for adversaries to affect the consumer device and in the worst case exfiltrate sensitive information.

In a perfect world, consumer-facing companies would be able to perfectly verify that all of the IP they include in their product was safe and correct. However, this may not be possible. For instance, IP providers want to keep their "secret sauce" to themselves, so they may encrypt the IP when sending it to other manufacturers. Additionally, with the proliferation of 3D die-stacking, it may be possible to integrate fully manufactured chips from third parties (which are very difficult to verify) into products.

This paper is the first paper in the architecture community to recognize the pitfalls of current system-accelerator protection mechanisms: they either provide full protection with low performance, or provide high-performance without full protection. We introduce a new protection mechanism, Border Control, that provides full protection and high performance with simple hardware. Instead of inventing a new protection specification from scratch, we leverage the current operating system protection specifications (processes and page tables), allowing the protection to be transparent to accelerator programmers.

This paper addresses an important threat from tightly integrated third-party IP: direct access to the shared physical memory. Even with trusted first-party design, it is difficult to ensure that hardware designs are completely correct, as demonstrated by processor errata. There have been previous cases where hardware bugs have been exploited to allow attacks such as privilege escalation [8]. By leveraging the differences between accelerators and CPUs (e.g., accelerators do not run the OS and therefore do not require access to all of memory), Border Control can provide added protection against accelerators, no matter their origin. Jerome Saltzer defined the *Principle of Least Privilege* by saying that "Every program and every user of the system should operate using the least set of privileges necessary to complete its job. Primarily, this principle limits the damage that can result from an accident or error." [9]. Although Saltzer was referring to software, these concepts can apply to hardware as well: rather than limiting the privileges of programs and users, we can limit the privileges of hardware components. Applying this principle may thus be a worthwhile aspiration when dealing with hardware accelerators.

This is a forward-looking paper that provides a solution to a future security threat. We make a number of assumptions based on extrapolations of current SoC trends. When looking back in five years, even if only a subset of our assumptions are true, we believe that this work will have a large impact. This paper may provide a catalyst for other researchers to find other interesting threat vectors in the emerging third-party accelerator economy, and to explore ways of mitigating and defending against these threats. Works like this, and other follow-on works, will pave the way for better security for consumer devices.

References

[1] *ARM Security Technology: Building a Secure System using TrustZone Technology*. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[2] "Risc-v." [Online]. Available: http://riscv.org/

[3] *AMD I/O Virtualization Technology (IOMMU) Specification, Revision 2.00*, Mar. 2011. [Online]. Available: http://support.amd.com/TechDocs/48882.pdf

[4] *ARM System Memory Management Unit Architecture Specification, SMMU architecture version 2.0*, 2012-2013. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ihi0062c/IHI0062C_system_mmu_architecture_specification.pdf

[5] *Coherent Accelerator Processor Interface User's Manual*, 2014. [Online]. Available: http://www.nallatech.com/wp-content/uploads/IBM_CAPI_Users_Guide.pdf

[6] *Intel Virtualization Technology for Directed I/O, Revision 2.3*, Oct. 2014. [Online]. Available: http://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/vt-directed-io-spec.pdf

[7] B. Black, "Die stacking is happening!" MICRO 2013 Keynote, Dec. 2013. [Online]. Available: http://www.microarch.org/micro46/files/keynote1.pdf

[8] M. T. Inc, *MIPS R4000PC/SC Errata, Processor Revision 2.2 and 3.0*, May 1994.

[9] J. Saltzer and M. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, Sept 1975.

[10] J. Stuecheli, B. Blaner, C. R. Johns, and M. S. Siegel, "CAPI: A coherent accelerator processor interface," *IBM Journal of Research and Development*, vol. 59, no. 1, pp. 7:1–7:7, Jan. 2015.