Poster: DMAArmor: Real-Time PCIe Packet Inspection for Enhanced DMA Security

Chenyang Li[1], Yuan Li[2], Wende Tan[3], Xinhui Han[1], Chao Zhang[4]
[1]Peking University, [2]Zhongguancun Laboratory, [3]Imperial College London, [4]Tsinghua University

***Abstract*—Direct memory access (DMA) devices pose severe threats to host systems. This paper presents a hardware-software co-designed bus-level defense solution, DMAArmor, which enforces fine-grained end-to-end DMA security by operating as a PCIe interposer. It comprises a pipeline to ensure real-time PCIe packet inspection and kernel module that instruments and extracts lifetime information of DMA buffers. Compared to existing IOMMU-based DMA defense approaches, DMAArmor provides spatial, temporal, and cross-device security enforcements for DMA access, without modifying the drivers or firmware. And DMAArmor extends protection to side channels and preboot stages, enabling full-lifecycle DMA security.**

I. Introduction

Direct Memory Access (DMA) is essential for high-performance I/O, but it also lets devices access memory without CPU mediation, making compromised or malicious devices powerful attackers. Existing host-based defenses [1], [2], [5], most notably IOMMUs, leave key gaps: page-granular permissions expose subpage regions, IOTLB-based revocation introduces TOCTOU windows, preboot lacks protection before the OS configures defenses, and host-side mechanisms have limited visibility into PCIe peer-to-peer (P2P) and protocol-level side channels. We present DMAArmor, an FPGA-based PCIe interposer that enforces programmable, bus-level DMA security by inspecting and filtering PCIe Transaction Layer Packets (TLPs) in real time. DMAArmor combines a match–action data plane with a lightweight OS control plane that derives DMA buffer lifetimes from the kernel DMA API and pushes per-buffer metadata and policy updates to hardware, enabling byte-granular spatial checks and immediate revocation without driver or SoC changes.

II. Background

a) DMA Abstraction: Modern OSes (e.g., Linux) provide a unified DMA API that drivers use to map/unmap DMA buffers. The mapping call `dma_map` specifies the buffer address and size, while the unmapping call `dma_unmap` ends the lifetime and should revoke access. It makes the DMA API a natural place to capture precise spatial and temporal semantics in a driver-agnostic way.

b) Threat Model: We assume the attacker controls at least one device (e.g., via malicious firmware, exploited vulnerabilities, or physical access) and can issue arbitrary DMA requests within PCIe protocol constraints to bypass existing defenses.

c) Limitations of Existing Defenses: Effective DMA defense should satisfy: (C1) byte-granular spatial safety, (C2) revocation aligned with `dma_unmap` to close TOCTOU windows, (C3) visibility into cross-device/P2P traffic, (C4) compatibility without driver/SoC modification, and (C5) low overhead on the DMA critical path. Current solutions typically trade off one or more of these dimensions [5], [6], [3], [4].

III. System Design

DMAArmor is an FPGA PCIe interposer deployed directly upstream of an untrusted device. It applies a match–action access-control pipeline to all TLPs entering/leaving the device, transparently forwarding benign traffic while parsing request/response headers (e.g., Requester ID, Tag, Address, and Length) to enforce policies in real time.

The data plane matches TLPs against programmable rules and then takes actions. At hardware-level, DMAArmor implements Match-Action Units (MAU) for policy enforcement. It classifies sensitive flows and takes corresponding actions (accept, log, modify, and drop). MAUs are evaluated in parallel and arbitrated with fixed priority.

A lightweight control plane instruments the OS DMA abstraction (`dma_map`/`dma_unmap`) to derive per-buffer lifetimes and pushes a per-buffer segment table plus policy updates to the FPGA via a PCIe BAR interface. This makes policy updates centralized and driver-agnostic. In particular, the segment table uses one entry per DMA buffer, so updating or revoking a buffer mapping is a single metadata write and does not require a TLB flush.

Policy enforcement. DMAArmor employs a match-then-act flow to check all PCIe TLPs that flows in and out from the targeted device. For DMA reads/writes, DMAArmor does bounds-checking with segment table. DMAArmor also supports P2P protection, preboot protection (blocking DMA and OpROM-related reads before policies are installed) and protocol-level sanitization for partial-write side channels.

A. Policy 1: Spatial and Temporal Safety

Because the `dma_map` API returns opaque pointers to the host driver and a specific device. These opaque pointers can be instrumented at kernel level. We carefully partition the device address space with high-bits (16-bits) representing the Segment ID (`SID`) and low-bits (32-bits) for the buffer offsets (`BO`). Meanwhile, the buffer lifetime information is extracted and sent to the DMAArmor via a PCIe BAR write.

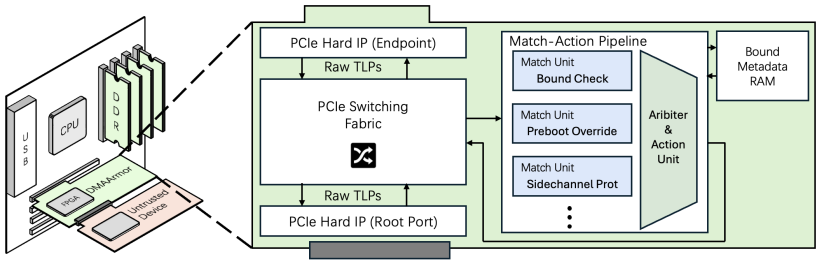


Fig. 1. System Overview of DMAArmor. All TLPs from and to the targeted device traverse the pipeline before forwarding.

Using this mechanism, the segment table uses one entry per buffer, holding segment start offset (`SSO`), segment end offset (`SBO`) and base address high (`BAH`). The segment table is indexed by segment id (`SID`), so atomic updates are made by just one write and no TLB flush is needed. After bounds checking, we substitute `SID` with `SBO`, so a total of 48-bits device addressable space is retained.

- Threat: Page-granular permissions allow subpage attacks, and IOTLB delays create TOCTOU windows.
- Match: Detect DMA read/write TLPs and split the address into Segment ID (SID) and Buffer Offset (BO). Use SID to look up the segment table and do range check for BO.
- Action: In-bounds reads are forwarded (with `SAH` translation); out-of-bounds reads return zeros. In-bounds writes are allowed; out-of-bounds writes are dropped.

Using policy 1, DMAArmor addresses C1 and C2 with byte-granular bounds checking and efficient segment revocation.

B. Policy 2: Cross-Device (P2P) Isolation

- Threat: If a PCIe switch is cascaded downstream of DMAArmor, EP-to-EP traffic could bypass DMAArmor entirely, rendering DMAArmor ineffective like other defenses.
- Match: Monitor CfgRd1/CfgWr1 packets, specifically those used to probe or enumerate downstream PCIe switches.
- Action: Block all Type 1 configuration requests to prevent insertion of downstream switches. This enforces a flat topology where any P2P traffic between endpoints must pass through DMAArmor, ensuring it remains subject to other policies. P2P DMA safety can be checked with Policy 1. We target a flat topology; supporting cascaded switches or richer P2P policies is future work.

C. Policy 3: Preboot Protection

BIOS-based defenses of preboot falls short in manageability. DMAArmor provides a hardware based mitigation, removing the need for manual configurations and BIOS updates.

- Threat: During boot, the OS hasn't configured the IOMMU yet, so devices can perform unrestricted DMA. BIOS/UEFI automatically loads device Option ROMs (OpROMs), letting attackers run malicious code before the OS takes control.
- Match: Detect DMA TLPs and configuration space reads targeting OpROM regions during preboot.
- Action: Block all DMA by default until the OS loads the first segment table. Hide OpROM presence. By returning *no driver* or *not available* for OpROM reads, delaying device initialization until security policies are in place.

D. Policy 4: Protocol-Level Side Channel Mitigation

PCIe, the main DMA protocol, is no longer just a data pipe but a complex protocol layer where implementations can leak information through side channels. DMAArmor mitigates CVE-2022-21166, device register partial write, by sanitizing unwanted data leakage within PCIe TLP packets.

- Threat: CVE-2022-21166 (Device Register Partial Write). PCIe TLPs use a `byte_enable` mask to indicate which bytes are actually written. On Intel processors, when writing to a non-aligned address, unused bytes in the TLP may unintentionally carry internal sensitive data.
- Match: Detect non-aligned register write requests and parse the `byte_enable` mask from the TLP.
- Action: Sanitize the write payload by zeroing out all bytes not covered by the byte_enable mask. This prevents leakage of sensitive data to untrusted devices while preserving correct protocol semantics.

IV. Conclusion and Future Work

DMAArmor enforces DMA security at the PCIe transaction layer using an FPGA based match-action engine. It provides byte-granular spatial and temporal protection, P2P isolation, and protocol level sanitization.

Future work: 1) test compatibility with a wide range of devices. 2) benchmark performance overhead on real-world workloads. 3) add support for cascaded switches

References

[1] I. AMD and O. Virtualization, "Technology (iommu) specification," 2007.
[2] N. Amit, M. Ben-Yehuda, and B.-A. Yassour, "Iommu: Strategies for mitigating the iotlb bottleneck," in *International Symposium on Computer Architecture*. Springer, 2010, pp. 256–274.
[3] A. Markuze, A. Morrison, and D. Tsafrir, "True iommu protection from dma attacks: When copy is faster than zero copy," in *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, 2016, pp. 249–262.
[4] A. Markuze, I. Smolyar, A. Morrison, and D. Tsafrir, "Damn: Overhead-free iommu protection for networking," in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 301–315.
[5] S. Peter, "Apple m1 dart iommu driver," 2021.
[6] X. Wang, W. Shen, Y. Bu, J. Zhou, and Y. Zhou, "DMAAUTH: A lightweight pointer integrity-based secure architecture to defeat DMA attacks," in *33rd USENIX Security Symposium (USENIX Security 24)*. Philadelphia, PA: USENIX Association, Aug. 2024, pp. 1081–1098.

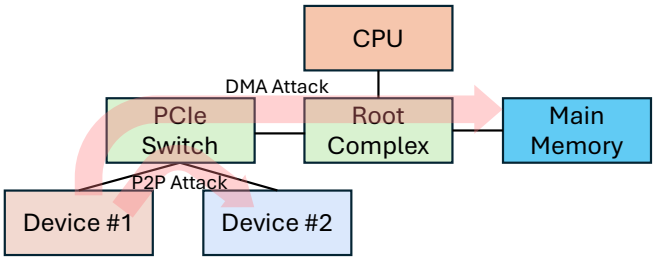
Poster: DMAArmor: Real-Time PCIe Packet Inspection for Enhanced DMA Security

Chenyang Li [1], Yuan Li [2], Wende Tan [3], Xinhui Han [1], Chao Zhang [4]
[1] Peking University, [2] Zhongguancun Laboratory, [3] Imperial College London, [4] Tsinghua University

1. Introduction

DMA enables high-performance I/O but gives devices direct memory access, creating a powerful attack surface. Traditionally, IOMMU-based defenses suffer from spatial and temporal gaps, limited preboot coverage, and weak protection for PCIe P2P traffic and protocol-side channels.

2. Background



Effective DMA protections have 5 critical criteria:

- C1 **Spatial safety**: byte-granularity checks against OOB attacks.
- C2 **Temporal safety**: To defend TOCTOU attacks, when *dma_unmap()* returns, the permission to access the buffer is simultaneously revoked.
- C3 **Cross-device isolation**: P2P DMA from one device to another must be monitored.
- C4 **Compatibility**: no driver or SoC modification.
- C5 **Performance**: minimal overhead in hardware and software.

3. Design

DMAArmor acts as an FPGA PCIe interposer upstream of untrusted devices. It features a match–action pipeline to inspect all PCIe Transaction Level Packets (TLPs). And it enforces programmable security policies at the boundary of untrusted devices.

4. Security Policies

Policy 1: Bound Check

- **Match:** MRd/MWr TLPs; parse Address/Length and match against bounds metadata table
- **Action:** zero-fill out-of-bounds reads; drop out-of-bound writes
- **Metadata**: DMA buffer boundaries are extracted from kernel DMA abstraction (*dma_map/unmap* functions)

Policy 2: Cross-Device (P2P) Isolation

- **Match:** CfgRd1/CfgWr1 TLPs
- **Action:** Drop TLP to prevent user insertion of downstream switches; ensure full monitoring of device

Policy 3: Preboot Override

- **Threats**: BIOS automatically loads device Option ROMs (OpROMs), leading to arbitrary code execution
- **Match**: DMA TLPs before policy install and host-initiated OpROM-related config-space reads.
- **Action**: Reject all DMA until Policy 1 is installed; block OpROM loads by returning no-op responses.

Policy 4: Side-channel Protection (CVE-2022-21166)

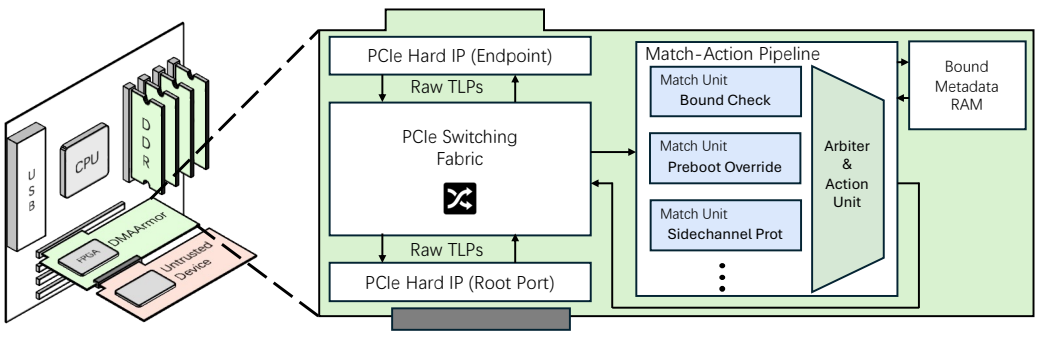
- **Match**: Non-aligned register writes; parse masks in write TLPs.
- **Action**: Mask payload bytes to zero where mask is unset, preventing leakage while preserving semantics.

5. Security Analysis

Threat Model

An attacker can control an untrusted device to launch DMA attacks or send any PCIe packets to spoof or construct side-channels.

- For C1 & C2, they are protected by Policy 1 at byte-granularity with immediate revocation.
- For C3, it is protected by Policy 2 to ensure full monitoring
- For C4, DMAArmor only instruments at kernel DMA abstraction layer and the PCIe bus, so no modifications are required for BIOS, Driver, or SoC hardware.
- For C5, performance is retained with hardware-based checks



6. Contribution / Takeaway

1. Byte-granular enforcement on PCIe TLPs via segment table (no page rounding)
2. Immediate revocation at dma_unmap (no IOTLB delay)
3. Pre-boot + Protocol Sanitization on bus (OpROM blocking + CVE payload masking)