# Hardware-assisted Memory Isolation

Hongyi Lu[*†]
luhy2017@mail.sustech.edu.cn
Research Institute of Trustworthy Autonomous Systems
Southern University of Science and Technology
China

## Abstract

Modern computing systems increasingly rely on hardware-assisted memory isolation to secure critical data and execution contexts without the overhead of purely software-based mechanisms. While features like Intel MPK, Arm POE, and RISC-V PMP offer promising support, they often suffer from a limited number of available isolation domains and primarily focus on CPU memory, leaving interactions with peripheral devices unprotected.

In this dissertation, we partially address these challenges by presenting Moat, a lightweight hardware-assisted isolation framework that combines MPK with the existing MMU to achieve effectively unlimited isolation domains, demonstrated in securing eBPF programs with minimal overhead. Beyond proposing new solutions, we also uncover a critical vulnerability in current GPU trusted execution environments (TEEs) through Mole, an attack that exploits under-documented microcontroller units inside GPUs to bypass the isolation of GPU TEEs and access protected memory.

Our research aims to develop new hardware primitives that are flexible, lightweight and able to deliver protection across CPUs and peripherals to meet the demands of secure, large-scale systems.

**ACM Reference Format:**
Hongyi Lu. 2025. Hardware-assisted Memory Isolation. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security (CCS '25), October 13–17, 2025, Taipei, Taiwan, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3719027.3765566

## 1 Introduction

As modern computer systems grow complex, ensuring their security via isolation has become challenging. Software-based isolation mechanisms, like software fault isolation (SFI) [24], are restricted by limitations such as overhead. This renders software solutions inefficient to be deployed in complex systems like OS kernels.

To address these limitations, researchers have resorted to hardware-based approaches to enforcing isolation [3–5, 9, 25]. These approaches leverage hardware features [3–5] to overcome the bottleneck of software-based solutions (e.g., slow permission checks). Conventional software-isolation mechanisms, such as software fault isolation (SFI), incur significant overhead that is proportional to the number of memory accesses performed. In contrast, hardware-assisted approaches can achieve isolation with much lower overhead. Due to this advantage, all the major platforms, including x86, Arm, and RISC-V, have introduced their own hardware features for flexible isolation, such as Intel's Memory Protection Keys (MPK) [21], Arm's Permission Overlay Extension (POE) [2], and RISC-V's Physical Memory Protection (PMP) [22].

On x86, MPK is primarily used for securing critical data structures (e.g., cryptographic keys) in memory. To better utilize MPK, Park et al. [21] proposed libmpk, which resolves the semantic gap between MPK and `mprotect` provided by the OS. One of the major shortcomings of MPK is that it only provides 16 isolation domains, which is insufficient for large-scale isolation. To solve this obstacle, VDom and EPK [10, 26] developed a scheme that uses Extended Page Tables (EPT) [13] to provide an unlimited number of protection keys. Apart from isolating user-space applications, MPK is also used for the isolation of trusted applications. SGXLock [8] establishes mutual distrust between the OS and SGX enclaves, while EnclaveDom establishes isolation within each enclave.

Unlike x86, which introduced MPK in 2015, Arm's POE is a relatively new feature released in 2022. Similar to MPK, POE also provides a limited number (up to 16) of permission IDs for isolating different memory regions. To address this limitation, Liu et al. [17] proposed NanoZone, which utilizes Arm's latest CCA to lift the restriction of available isolation domains.

RISC-V's PMP takes an unusual design compared to MPK and POE. Instead of storing protection keys inside the Page Table Entries (PTEs) and isolating at virtual memory level, PMP stores the isolation configuration (e.g., address range) in a dedicated register and directly isolates at physical memory level. This makes PMP a more fundamental isolation mechanism that can be used at low-level system components, such as hypervisors. However, this also restricts the flexibility of PMP. For example, if a domain switch occurs, the software must trap into the system monitor to obtain permission to reconfigure the PMP. To address this issue, supervisor PMP (sPMP) [23] was proposed. sPMP allows the OS kernel at supervisor-level to manage the permissions of isolation domains, reducing the overhead. PMP is also widely used in various scenarios that require isolation. For example, Kuhne et al. [15] proposed Dorami to isolate untrusted vendor firmware from system monitors. Keystone [16] re-purposes PMP as a security primitive to support Trusted Execution Environment (TEE). Despite all these applications, PMP faces the same restriction on the number of available isolation domains similar to MPK and POE. PMP typically provides only 8 domains, which is insufficient for large-scale isolation.

There are also a series of hardware features that are specially designed for TEEs, such as Intel Software Guard eXtensions (SGX) [20],

[*]Also with Department of Computer Science and Engineering, Southern University of Science and Technology.
[†]Also with Department of Computer Science and Engineering, Hong Kong University of Science and Technology.


*CCS '25, October 13–17, 2025, Taipei, Taiwan, China*

ACM ISBN 979-8-4007-1525-9/2025/10
https://doi.org/10.1145/3719027.3765566

Arm TrustZone [5]. These features are designed to provide a dedicated secure execution environment (often referred to as *enclave*) for sensitive applications. For example, both SGX and TrustZone allow developers to mark certain memory pages as *secure*, forbidding the untrusted OS from accessing them. Recently, the concept of Confidential Virtual Machine (CVM) was proposed to extend the TEE concept to virtual machines. CVM allows hosting virtual machines in a secure enclave, which is isolated from the untrusted hypervisor and OS. All major platforms have introduced their own CVM implementations, such as Intel Trust Domain eXtension (TDX) [14] and Arm Confidential Compute Architecture (CCA) [6]. Unlike lightweight primitives like MPK, which focus on memory isolation, these features provide a complete isolation solution that includes memory, interrupt, and I/O, making them suitable for VM isolation.

The above-mentioned hardware features are primarily CPU-oriented, which means they are designed to isolate memory regions in the CPU address space; peripheral devices are *not* in the scope of these hardware features. To defend against attacks launched from peripheral devices, such as DMA attacks, the I/O Memory Management Unit (IOMMU) [1] was proposed. IOMMU is a hardware component that is similar to CPU MMU. It maintains a page table structure to translate peripherals' virtual addresses to physical addresses. Recent works [9, 12, 25] such as StrongBox combine IOMMU with traditional CPU-side isolation primitives to build TEE for GPUs. Though IOMMU is effective in mitigating attacks targeting peripheral memory, the lack of unified isolation mechanism does not fully consider the intricacy of modern computer systems, which leads to potential security issues.

**Summary.** From the above discussion, we can observe that one of the most significant limitations of existing hardware-assisted isolation mechanisms is the scarcity of hardware resources. Currently, all the major platforms only provide hardware features with a limited number of isolation domains up to 16, which is insufficient for large-scale isolation scenarios. Moreover, though works like VDom [10, 26] extend this limit by incorporating other hardware features, they rely on heavyweight hardware features like EPT, reducing the flexibility of such lightweight isolation mechanisms. Aside from the limit of isolation domains, a critical yet often overlooked aspect is that most of these hardware isolation primitives are designed purely for CPU memory isolation. Though peripheral isolation schemes like IOMMU have been proposed, the intricate interaction between CPU and peripheral devices is often ignored. This leads to a lack of comprehensive isolation solutions that can effectively secure both CPU and peripheral memory spaces.

## 2 Research Questions

**RQ1.** *How can we address the scarcity of hardware isolation domains without heavyweight hardware extension?*

The new extension should not impair the flexibility and performance of the original hardware features. To do so, it should not introduce other heavyweight hardware features like EPT or CCA, which require complicated setup and impose additional overhead.

**RQ2.** *What kind of security issues can arise from the lack of unified isolation mechanism between CPU and peripheral devices?*

To answer this question, we analyze the existing security solutions used for peripherals and search for the potential security issues that emerge from the interaction between the CPU and the peripherals.

## 3 Preliminary Results

**Results on RQ1.** We have answered the first research question, by proposing Moat [19], a hardware-assisted isolation mechanism targeting extended Berkeley Packet Filter (eBPF) that relies solely on the MPK and off-the-shelf MMU. Moat achieves several key objectives. First, it provides a virtually unlimited number of isolation domains with only 16 MPK entries. Secondly, Moat is designed to be lightweight, it introduces a less than 10% overhead even with hundreds of concurrent isolation domains. This is achieved via a novel two-layer isolation scheme that combines both MPK and MMU. In the first layer, we leverage MPK to isolate components that require frequent switching, such as core kernel and eBPF programs; MPK ensures that these components can be switched efficiently. In the second layer, we utilize MMU to build up isolation between different eBPF programs. To avoid the overhead brought by the MMU, we deploy an emerging hardware feature called Process Context Identifier (PCID). By assigning PCIDs to different eBPF programs, Moat switches eBPF programs without the costly TLB flushes. Though the implementation of Moat focuses on the isolation of eBPF programs, there are no significant obstacles that stop Moat from being applied to other cases. A similar design is later adopted by other researchers to isolate loadable kernel modules [11].

**Results on RQ2.** We have some preliminary results on the second research question as well. When analyzing the security of peripherals, we noticed that current isolation primitives often overlook the interaction between the CPU and the peripherals and the internal complexity of modern hardware, such as GPUs. Based on the observation, we developed Mole [18], an attack targeting existing GPU TEEs on Arm. Specifically, we find that GPU isolation solutions like CAGE [25] and StrongBox [9] adopt a shim-style design; it delegates certain non-critical routines to the untrusted world to reduce the size of Trusted Code Base (TCB), such as GPU initialization and power management. Since these non-critical routines neither process any sensitive data nor chronologically overlap with the execution of the isolated GPU tasks, this delegation is considered safe by these GPU TEEs. However, we find that the modern Arm GPU embeds a new component called Command Stream Frontend (CSF) [7]. CSF is an under-documented Microcontroller Unit (MCU) that is responsible for offloading GPU scheduling from the CPU. As an embedded unit, CSF's memory access is equivalent to the GPU's memory access, which means that CSF can access *any* memory regions that the GPU can access. To make the matter worse, in the design of these shim-style GPU TEEs, the GPU initialization is delegated to the untrusted world. The untrusted OS can easily inject a malicious firmware into the CSF and use it as a trampoline to access the secure GPU memory, bypassing all the isolation mechanisms deployed, such as IOMMU, TrustZone. Though shim-style GPU TEEs are primarily academic prototypes, industry involvement (CAGE/StrongBox is contributed by authors from Alibaba) suggests future production deployment. Mole unveils the potential security issues that may arise from this design.

In addition to GPU, this design is also frequently explored by researchers to protect other peripherals, such as USB. Therefore, we emphasize that the security risk exposed by Mole should not be underestimated. Lastly, solving vulnerabilities like Mole requires a security solution that takes the peripherals into consideration.

## 4 Future Work

In the future, we aim to explore both **RQ1** and **RQ2** by analyzing the limitations of current hardware-assisted isolation. We plan to conduct a systematic survey on the existing hardware security features. Based on the survey, we will leverage existing features and design new ones to achieve the following goals in the future.

**Lightness and flexibility.** Two of the most important objectives of our new isolation scheme is lightness and flexibility. Though Moat has partially achieved this goal using MMU, it requires the workload to have specific properties like eBPF. Such properties might not hold in practice. We plan to enhance the design of Moat with a layered isolation structure like MMU. However, the MMU on modern platforms is too complex. We will investigate the features of real-world workloads and debloat our design to make it suitable for real-world applications without compromising its lightness.

**Whole-system security.** Another important objective of our new isolation design is to take peripherals into consideration. Due to the intricacy of modern hardware, many peripherals now also have their own "CPU" and "OS" (i.e., MCU and firmware), which is viewed as a part of the peripherals and can be used to launch attacks against OS or TEE. Since these components often lack implementation details and documentation, it is non-trivial to design an isolation mechanism for them at the hardware level. Fortunately, these peripherals rely on a unified interface `/lib/firmware` in the Linux kernel to load the firmware. Therefore, our next step is to systematically investigate the firmware and try to develop an isolation scheme that can effectively mitigate the associated risks.

## 5 Conclusion

We present a summary of our dissertation, where we design a hardware-assisted isolation solution for eBPF programs and unveil an imminent risk from the under-documented MCUs in modern peripherals. We aim to design new hardware isolation primitives that are lightweight, flexible, and able to mitigate the threats from peripheral devices. We have highlighted our motivation for undertaking this research, the importance of hardware isolation, and potential risks of existing works. We have identified two key research questions and provided preliminary results on both of them. Moving forward, our remaining work involves further exploring these two directions and designing better isolation primitives.

## Acknowledgments

I would like to thank my advisor Prof. Fengwei Zhang and Prof. Shuai Wang for their continuous support and guidance. This work is partly supported by the National Natural Science Foundation of China under Grant No. 62372218 and No. U24A6009.

## References

[1] AMD. 2025. AMD I/O MMU Specification. https://instinct.docs.amd.com/projects/amdgpu-docs/en/latest/conceptual/iommu.html

[2] Arm. [n. d.]. Arm Permission indirection and permission overlay extensions. https://developer.arm.com/documentation/102376/0200/Permission-indirection-and-permission-overlay-extensions.

[3] Arm. 2024. Arm Confidential Compute Architecture. https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture

[4] Arm. 2024. Stage 2 translation. https://developer.arm.com/documentation/102142/0100/Stage-2-translation

[5] Arm. 2024. TrustZone for Cortex-A. https://www.arm.com/technologies/trustzone-for-cortex-a

[6] Arm. 2025. Arm Confidential Compute Architecture. https://www.arm.com/architecture/security-features/arm-confidential-compute-architecture

[7] Boris Brezillon. 2024. PanCSF: A new DRM driver for Mali CSF-based GPUs. https://www.collabora.com/news-and-blog/news-and-events/pancsf-a-new-drm-driver-for-mali-csf-based-gpus.html

[8] Yuan Chen, Jiaqi Li, Guorui Xu, Yajin Zhou, Zhi Wang, Cong Wang, and Kui Ren. 2022. SGXLock: Towards Efficiently Establishing Mutual Distrust Between Host Application and Enclave for SGX. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 4129–4146.

[9] Yunjie Deng, Chenxu Wang, Shunchang Yu, Shiqing Liu, Zhenyu Ning, Kevin Leach, Jin Li, Shoumeng Yan, Zhengyu He, Jiannong Cao, et al. 2022. Strongbox: A gpu tee on arm endpoints. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 769–783.

[10] Jinyu Gu, Hao Li, Wentai Li, Yubin Xia, and Haibo Chen. 2022. EPK: Scalable and Efficient Memory Protection Keys. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*. USENIX Association, Carlsbad, CA, 609–624.

[11] Yinggang Guo, Zicheng Wang, Weiheng Bai, Qingkai Zeng, and Kangjie Lu. 2025. BULKHEAD: Secure, Scalable, and Efficient Kernel Compartmentalization with PKS. In *32nd Annual Network and Distributed System Security Symposium, NDSS 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society. https://www.ndss-symposium.org/ndss-paper/bulkhead-secure-scalable-and-efficient-kernel-compartmentalization-with-pks/

[12] Seung-Kyun Han and Jinsoo Jang. 2023. MyTEE: Own the Trusted Execution Environment on Embedded Devices.. In *NDSS*.

[13] Intel. [n. d.]. 5-Level Paging and 5-Level EPT White Paper. https://www.intel.com/content/www/us/en/content-details/671442/5-level-paging-and-5-level-ept-white-paper.html.

[14] Intel. 2025. Intel Trust Domain eXtension. https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html

[15] Mark Kuhne, Stavros Volos, and Shweta Shinde. 2024. Dorami: Privilege Separating Security Monitor on RISC-V TEEs. arXiv:2410.03653 [cs.CR] https://arxiv.org/abs/2410.03653

[16] Dayeol Lee, David Kohlbrenner, Shweta Shinde, Krste Asanović, and Dawn Song. 2020. Keystone: an open framework for architecting trusted execution environments. In *Proceedings of the Fifteenth European Conference on Computer Systems* (Heraklion, Greece) *(EuroSys '20)*. Association for Computing Machinery, New York, NY, USA, Article 38, 16 pages. doi:10.1145/3342195.3387532

[17] Shiqi Liu, Yongpeng Gao, Mingyang Zhang, and Jie Wang. 2025. NanoZone: Scalable, Efficient, and Secure Memory Protection for Arm CCA. arXiv:2506.07034 [cs.CR] https://arxiv.org/abs/2506.07034

[18] Hongyi Lu, Yunjie Deng, Sukarno Mertoguno, Shuai Wang, and Fengwei Zhang. 2025. MOLE: Breaking GPU TEE with GPU-Embedded MCU. (2025).

[19] Hongyi Lu, Shuai Wang, Yechang Wu, Wanning He, and Fengwei Zhang. 2024. MOAT: Towards Safe BPF Kernel Extension. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, 1153–1170. https://www.usenix.org/conference/usenixsecurity24/presentation/lu-hongyi

[20] Frank McKeen, Ilya Alexandrovich, Ittai Anati, Dror Caspi, Simon Johnson, Rebekah Leslie-Hurd, and Carlos Rozas. 2016. Intel Software Guard Extensions (Intel SGX) Support for Dynamic Memory Management Inside an Enclave. In *Proceedings of the Hardware and Architectural Support for Security and Privacy 2016* (Seoul, Republic of Korea) *(HASP '16)*. Association for Computing Machinery, New York, NY, USA. doi:10.1145/2948618.2954331

[21] Soyeon Park, Sangho Lee, Wen Xu, HyunGon Moon, and Taesoo Kim. 2019. libmpk: Software Abstraction for Intel Memory Protection Keys (Intel MPK). In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*. USENIX Association, Renton, WA, 241–254.

[22] RISC-V. [n. d.]. RISC-V PMP. https://riscv.org/specifications/ratified/.

[23] RISC-V. [n. d.]. RISC-V sPMP. https://github.com/riscv/riscv-spmp.

[24] Gang Tan. 2017. . doi:10.1561/3300000013

[25] Chenxu Wang, Fengwei Zhang, Yunjie Deng, Kevin Leach, Jiannong Cao, Zhenyu Ning, Shoumeng Yan, and Zhengyu He. 2024. CAGE: Complementing Arm CCA with GPU Extensions. In *Network and Distributed System Security (NDSS) Symposium*.

[26] Ziqi Yuan, Siyu Hong, Rui Chang, Yajin Zhou, Wenbo Shen, and Kui Ren. 2023. VDom: Fast and Unlimited Virtual Domains on Multiple Architectures. In *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2* (Vancouver, BC, Canada) *(ASPLOS 2023)*. Association for Computing Machinery, New York, NY, USA, 905–919. doi:10.1145/3575693.3575735