Secure Domain-Specific Debugging on an MCU

Alvin Che-Chia Chang and Paul Shan-Chyun Ku

Andes Technology Corporation

Abstract

A modern embedded system involves multiple developers with varying security requirements, often leading to trust issues and the need for isolated proprietary assets. A secure monitor addresses this by providing multiple logically isolated execution environments (EEs), each maintaining its own state, ensuring invisibility between EEs. Traditional debugging mechanisms in RISC-V allow unrestricted access, compromising security by bypassing runtime isolation measures. Existing secure debugging methods authenticate debuggers but grant full access upon successful authentication, undermining the intended isolation. This talk presents a domain-specific debugging approach that restricts debugger access to predefined assets based on authentication, maintaining security and isolation during debugging. The solution is implemented into our secure monitor without using the RISC-V Debug Module, resulting in a low-cost, highly secure, and flexible debugging environment.

Introduction

A modern embedded system usually goes through a series of developers, from IP providers, SoC integrators, chip vendors, and firmware developers to OEMs and ODMs. When multiple entities and developers collaborate on a single system, often with differing security requirements, they may not trust one another and need their proprietary security domains. Thus, it's essential to isolate their private assets and system resources during run time as well as debug time. In a secure system, the job falls on the shoulder of its secure monitor. It shall provide multiple logically isolated execution environments (EE), each with its own state, including memory sections with specific permission, an interrupt handler, and a register file. The state of an EE should be deemed invisible to another EE except specifically expressed. It is held not only for run time but debug time. This talk will focus on building such secure debugging.

Developers often require debugging capabilities to identify and resolve system issues when developing domain applications. However, due to the unrestricted nature of RISC-V debug mechanisms, one can access anything without limitation via a debugger. The legacy secure debugging asks for authentication to control the debugger. As a result, the one getting authentication can access everything. It breaks the isolation people build for run time. Thus, domain-specific debugging becomes crucial, which means a system can define debuggable assets according to different authentications.

This presentation will demonstrate a secure domain-specific debugging solution, which not only provides comprehensive debugging capabilities but also addresses the aforementioned supply chain vulnerabilities that can compromise system security during debugging. We implemented such a secure monitor without using the RISC-V Debug Module and delivered a low-cost, highly secure, and highly flexible debugger.

The Secure Monitor

We will present secure domain-specific debugging by demonstrating our RISC-V base secure monitor designed to secure multiple domains in an MCU-class processor. The secure monitor runs in M-mode and manages security domains for trusted and untrusted applications in U-mode. It ensures application isolation using RISC-V security primitives, including PMP/Smepmp [2] and IOPMP [3].

Secure Domain-specific Authentication

To ensure the integrity between developers at every stage, the secure monitor incorporates asymmetric digital signatures to verify each security domain. Each security domain can be signed by its keypair, which belongs to the corresponding developer. Each security domain must be verified before being executed. Besides, the secure monitor will check the conflict and irritation of resource allocation. During run time, it can also deal with the resource sharing between domains in stack and dynamic ways. The sharing mechanism emulates the CHERI-style compartment [4] by using PMP/Smepmp to ensure every memory block being shared without creating unexpected access rights. To sum up, the secure monitor enforces domains' isolation and provides a certain level of flexibility to avoid copying data by a carefully designed sharing mechanism.

RISC-V Debug

Debugging is a crucial step in system development, enabling developers to identify and resolve issues, optimize performance, and ensure system robustness. The RISC-V standard defines two primary debugging scenarios: native debug (also known as self-hosted debug) and external debug. Native debug relies on debug software running on

the RISC-V platform to debug other components within the same platform, whereas external debug depends on a hardware-based debug module. In general, the native debug software operates in a higher privilege mode (e.g., OS or secure monitor) to access lower-privileged modes for debugging. The Sdtrig (Trigger Module, TM) ISA extension, defined in the RISC-V debug specification, facilitates native debug operations. To communicate with the host debugger, the native debug software receives and executes debugging commands to debug the system, utilizing the TM CSRs to perform operations such as inserting hardware breakpoints and instruction stepping.

Methodologies

Secure Domain-Specific Debugging

The secure monitor provides secure domain-specific debugging by leveraging the RISC-V native debug mechanism. It doesn't need the RISC-V Debug Module (DM), which can access every corner without limitation. Instead, a dedicated channel is integrated to facilitate communication between the host debugger and the target system. The channel can be as simple as a UART and should only be manipulated by the secure monitor (M-mode). PMP/Smepmp plus IOPMP can ensure the requirement.

Challenge-Response Authorization

The secure monitor authenticates and identifies the host debugger through a challenge-response authentication protocol. When a debugger connection is required from the host side, it generates challenging random data and sends it back to the debugger. Then, according to the keypairs and the challenging data, the debugger sends responding data to the secure monitor.

By leveraging the authentication keys of each security domain, the secure monitor verifies the response data to pick up debuggable EEs.

Debug Operations

Upon successful authorization, the secure monitor begins receiving and executing debugging commands from the host debugger. It processes incoming debug commands to facilitate debugging within the targeted security domain. The Trigger Module (TM) registers are managed by the secure monitor to perform targeted debug operations such as inserting hardware breakpoints or instruction stepping. Some operations involve memory addresses and sizes, i.e., memory regions. For instance, the host debugger may request reading or writing specific memory contents or insert a breakpoint at a particular instruction address. To ensure domain isolation, the secure monitor scrutinizes every debug operation involving memory regions, verifying that the host debugger only accesses permitted memory regions. By doing so, the secure monitor restricts the host debugger's access to authorized assets within the protection boundary of the host debugger's security domains.

PMP, Smepmp, IOPMP

Notably, Smepmp is designed to prevent higher-privileged software from accessing or executing memory regions of less-privileged modes, thereby preventing unintended accesses. However, debug operations require the secure monitor to temporarily access memory content within the security domains, posing a challenge to Smepmp. To mitigate this issue, the secure monitor reserves high-priority PMP entries, and configures the entries that cover targeted memory regions temporarily for debugging, ensuring seamless debug operations while maintaining security.

When debugging memory addresses that target Memory-mapped I/O (MMIO), the debug operations may manipulate peripherals like the Direct Memory Access (DMA) controller and request DMA to access system memory. However, PMP and Smepmp, which provide standard protection schemes for RISC-V hart accesses to physical address space, do not safeguard against unauthorized accesses from non-hart initiators such as DMA controllers. This creates a vulnerability where a host debugger can initiate debug operations that manipulate the DMA controller to access out-of-bounds memory regions. To mitigate this risk, IOPMP is integrated into the system. As a hardware component, IOPMP safeguards data by denying unauthorized accesses from I/O agents, much like PMP and Smepmp restrict RISC-V hart accesses. Adopting IOPMP can ensure that memory regions accessible by non-hart initiators are restricted. Even when a host debugger initiates debug operations to start DMA transactions, IOPMP restricts these transactions to accessing only memory regions within the debugger's security domains.

Remarks

We introduced our secure monitor, which not only constructs multiple EEs in run time but in debug time. It uses as few hardware components as possible to achieve the ability to optimize the cost, and by its software nature, the debug ability can be removed or enhanced via a firmware upgrade. It further improves security after the deployment.

References

[1] "RISC-V Debug Specification v1.0.0-rc4", github.com/riscv/riscv-debug-spec.

[2] "The RISC-V Instruction Set Manual Vol. II: Privileged Architecture", github.com/riscv/riscv-isa-manual.

[3] "RISC-V IOPMP Specification v0.9.2-rc3", github.com/riscv-non-isa/iopmp-spec.

[4] "RISC-V Specification for CHERI Extensions v0.9.3-prerelease", github.com/riscv/riscv-cheri.