HEX FIVE MULTIZONE® SECURITY FAQ for Arm® Platforms

What is MultiZone Security?

It is a software defined, hardware enforced Trusted Execution Environment (TEE) for embedded systems. It's a safe and easy alternative to Arm® TrustZone® on Cortex®-M0+/M3/M4/M7.

What is the breakthrough of MultiZone Security?

- Tiny attack surface: ~4KB. Formally verifiable
- Software defined, policy driven. No redesign required
- Secures entire system including programs, data, I/O, IRQs
- Can retrofit existing hardware

What are the threat models of MultiZone?

Threat models MultiZone address are remote attacks, supply chain risks, DoS. On an embedded system, critical functions such as root of trust, chain of trust, network stack, control functions for actuators etc should be shielded in their own zones to improve the safety, security and availability of the system.

Refer to examples:

https://github.com/hex-five/multizone-sdk-arm

https://github.com/hex-five/multizone-secure-iot-stack

Is MultiZone Security hardware or software?

It is both! MultiZone Security is hardware-enforced, software-enabled security through separation. The necessary hardware primitives are already built-in to the cores. The software provided by Hex Five is the enabler for these hardware primitives.

What are MultiZone hardware requirements for Arm?

MultiZone runs on Cortex-M0+/M3/M4/M7 to leverage the MPU (memory protection unit), operational modes and privilege levels. It secures the entire system including programs, data, I/O and IRQs.

Where do I start?

To get you started, we have developed a few fully functional reference applications based on free and open hardware and software components. All FPGA bitstreams, software and documentation are freely available online at https://github.com/hex-five, Hex Five's GitHub repository

You may want to start with GitHub https://github.com/hex-five/multizone-sdk-arm which has reference implementation on several Arm platforms.

Does MultiZone secure a SoC that includes both Cortex-M and Cortex-A cores?

Yes! MultiZone can secure the whole SoC. The applications that are on the A cores can be separated from critical and trusted functions on the M cores, resulting in a much stronger security posture and higher system availability.

Is MultiZone Security free?

Yes! The https://github.com/hex-five/multizone-api Multi Zone API definition is free as in "free speech" and free as in "free lunch". It is a free and open standard open sourced under ISC permissive license (https://github.com/hex-five/multizone-api/blob/master/LICENSE") for "any use". It is maintained by Hex Five and we welcome everyone's direct contributions in the form of a GitHub pull request. Everyone can write their own implementations of the free and open API. We are strong supporter of the commercial open source movement and actively promote the development of a global vibrant community!

The MultiZone Security commercial implementation provided by Hex Five includes MultiZone Configurator and MultiZone Runtime. These components are distributed on GitHub (https://github.com/hex-five/multizone-sdk-arm) free of charge for evaluation only. Commercial use is restricted and requires Hex Five commercial license, sometime included with the hardware itself, royalty-free. Hex Five's core components are suitable for formal verification and everyone is welcome to perform formal verification and to publish models and results. In addition, Hex Five's source code is always available to commercial licensees for inspection and detailed code review in a clean room environment according to the term of the commercial license.

What's the difference between the free MultiZone version on GitHub and the commercial one?

The only difference is the license. Code and functionality are the same.

The commercial license entitles Hex Five's customers to:

- commercial use
- patent protection and indemnification
- clean room environment access to source code review
- hot line technical support
- early access to new features and security fixes

Is MultiZone configuration hardcoded?

No, it is not.

Under the terms of the evaluation license anyone can modify zones' code and security configuration.

https://github.com/hex-five/multizone-sdk/blob/master/LICENSE

Is multizone.jar the item HexFive protects in the free version?

All Hex Five Intellectual Propriety is protected domestically and internationally: patent pending US 16/450,826 PCT US19/38774 [62/689,205 62/796,967 62/796,991 62/801,091].

Except for the MultiZone Runtime, all Hex Five source code is published on GitHub at https://github.com/hex-five

Is the MultiZone toolchain extension multizone.jar open and do we get the sources as well?

The toolchain extension mutizone.jar is not open source.

The output generated is 100% compliant with the free and open standard multi zone API.

Anyone can build their own implementation or license Hex Five's commercial implementation.

Hex Five's commercial licensees - as in paying customers - are entitled to clean room review of all Hex Five's source code under the terms of the commercial license.

What exact HEX format is the MultiZone toolchain extension multizone.jar outputting?

100% standard Intel Hex format - see https://en.wikipedia.org/wiki/Intel_HEX

What kind of support do you provide to help prospects develop their Proof Of Concept?

We offer the MultiZone Pilot Program: it is optional and intended to reduce customer's development time, cost and risk.

The MultiZone Pilot Program offers 40 hours of technical support over a period of 3 months for:

- hot line direct access to MultiZone development team to address any technical questions
- nonrecurring engineering work to adapt core product functionality to specific customer requirements
- The cost of the program is minimal and deductible from the cost of the first license

How do I debug MultiZone based applications?

In the very same way you debug any other Arm applications!

There is nothing secret, hidden, or special about MultiZone software, so no proprietary tools are required. MultiZone Security is 100% compliant with the Arm ISA. We believe in open source and use free and open tools for our internal development: GNU toolchain, GDB, openOCD, Eclipse CDT IDE or plain old school command line tools. But you can certainly use any other tools of choice as long as they work with Arm.

I've been writing systems software for 30 years, and I couldn't understand what this actually does, what the APIs are, and how you don't have to rewrite software. What's the performance hit? Where's the proof that ``no changes required``? Show me some standard software running on this.

Don't feel bad! MultiZone (https://hex-five.com/multizone-security-for-arm-cortex-m/) is such a revolutionary concept that it usually takes more than a minute or two to grasp. It is based on the state-of-the-art philosophy of zero-trust computing and security through separation - like Docker at the chip level.

Here is how it works from a development flow perspective

- break your current monolithic design in a few separate functional blocks - i.e OS with user apps, connectivity stack, root of trust, crypto libraries, bare metal drivers for DMA, I/O, etc
- define messages and listeners for inter-zone communications as a secure and modern way to expose and consume microservices
- code, compile, debug and test each individual component separately using programming language, toolchain and development environment of choice

- write plain text MultiZone Security policies with your text editor of choice, perhaps following the examples and the comprehensive documentation provided with the MultiZone SDK
- run the MultiZone Configurator, which generates a single signed firmware image including your original binaries, the MultiZone Runtime, and the security policies
- flash, run and debug as usual. And perhaps use the dummy terminal app provided with the SDK to verify and test security policies and system separation.

Be the hero of the day. You just made the world a better - and safer - place!

What are the APIs?

The MultiZone API (https://github.com/hex-five/multizone-api) is optional and provided as a linkable C library wrapper, but you could use plain inline assembly if you prefer. These are exception-based helper functions to interact with the MultiZone runtime - a bit like Linux system calls. As with system calls, you need them just for very specific things such as sending and receiving messages, to speed up the emulation of privileged instructions, to register traps and interrupt handlers and to set soft-timers.

How come you don't have to rewrite software?

Good application software usually does not interact directly with the hardware layer. So your apps are likely to work just fine without any modifications. System level software, such as FreeRTOS, is designed to work at the highest level of privilege and requires some minor changes to run in a secure unprivileged mode. However, MultiZone Security includes transparent support for trap & emulation of privileged instructions, so even this kind of code will run without major changes. In any case, Hex Five has already done the work for you and published a complete set of unprivileged, thus secure, versions of these software components. All open source and freely available on GitHub

What's the performance hit?

CPU throughput overhead is negligible. Typical context switch is in the order of 100 instructions and, depending on the implementation, it can take anything in between 120 to 180 cycles - you run the math but that's basically nothing. Interrupt latency on the other hand may take a few extra instructions if the Zone mapped to the interrupt is not in scope. A good use of the ECALL_YIELD() API can greatly minimize this effect. And if interrupt latency is really a requirement, there are ways to design the system separation so that the Zone associated with high priority interrupts is almost always in scope or ready to wake up from the Waiting For Interrupt state (WFI).