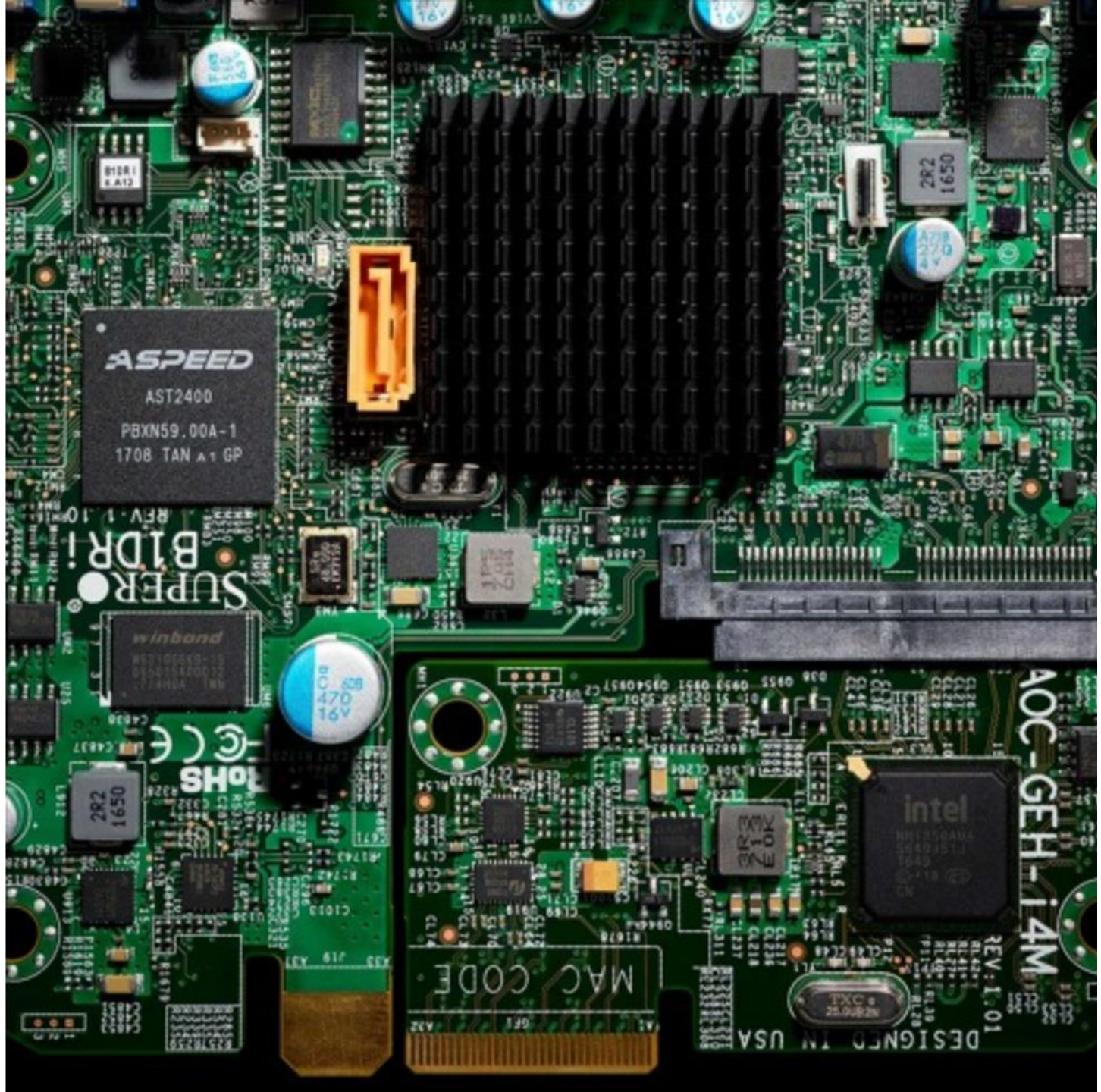
HEX-Five™

Making RISC-V the Most Secure Platform Ever

Cesare Garlati, Hex Five Security

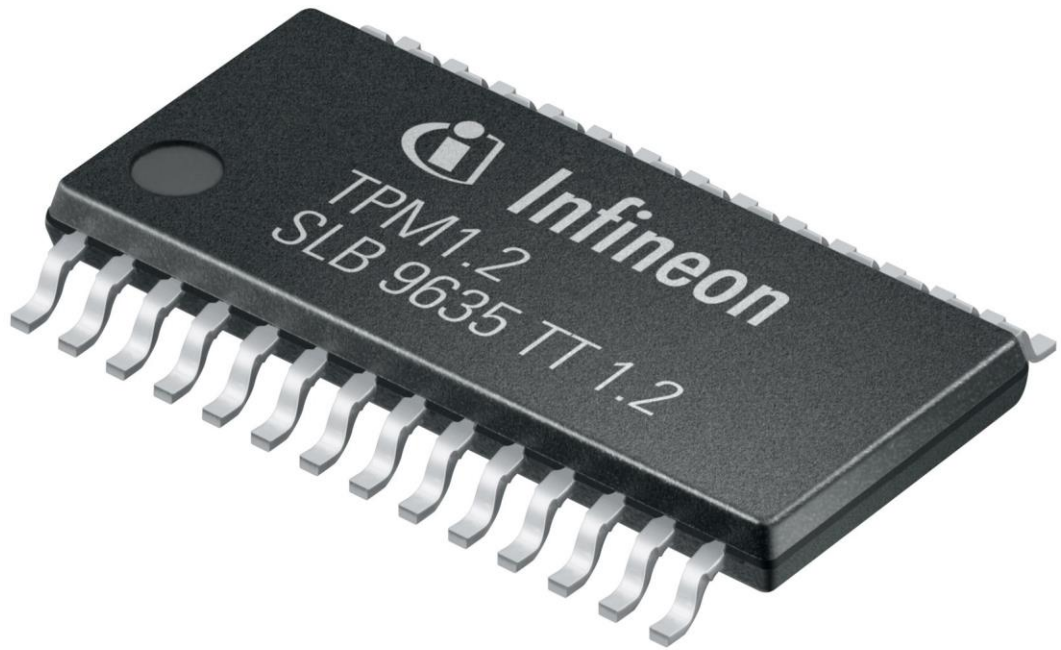
RISC-V Summit – December 4, 2018


ASPEED
AST2400
PBXN59.00A-1
1708 TAN A1 GP
SUPER B1DR!
winbond
intel
MAC CODE
AOC-GEH-i4M
REV:1.01
DESIGNED IN USA

Bloomberg Businessweek
October 8, 2018

The Big Hack

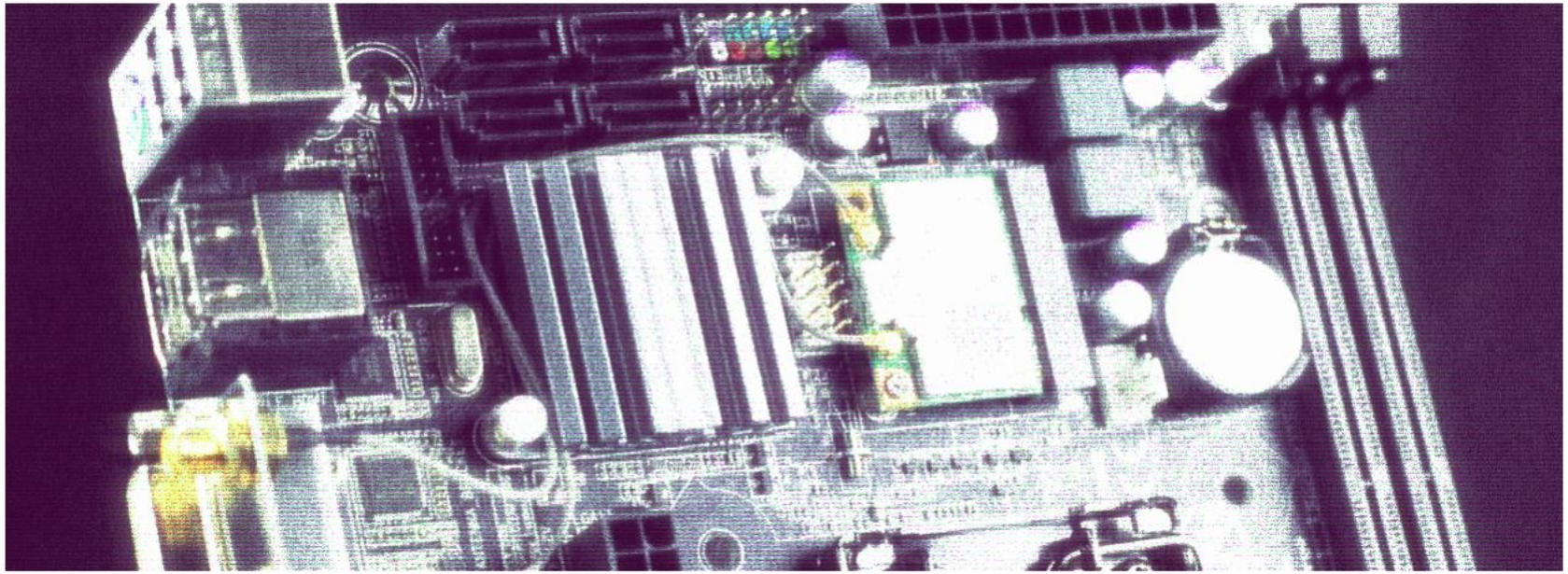
How [redacted] used a tiny chip to infiltrate America's top companies


Infineon
TPM1.2
SLB 9635 TT 1.2

Researchers Detail Two New Attacks on TPM Chips

By Catalin Cimpanu

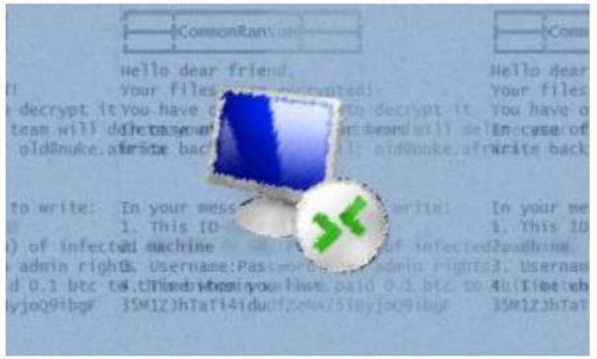
August 29, 2018 02:03 PM 1



Some PC owners may need to apply motherboard firmware updates in the near future to address two attacks on TPM chips detailed earlier this month by four researchers from the National Security Research Institute of South

POPULAR STORIES

New Stuxnet Variant Allegedly Struck Iran



CommonRansom Ransomware Demands


intel
SGX
Tool

SgxPectre attack allows to reveal the content of the SGX enclave

March 5, 2018 By Pierluigi Paganini

A group of researchers from the Ohio State University has discovered a new variation of the Spectre attack named SgxPectre that allows to reveal the content of the SGX enclave.


free
RTOS

FreeRTOS TCP/IP Stack Vulnerabilities Put A Wide Range of Devices at Risk of Compromise: From Smart Homes to Critical Infrastructure Systems

zLabs | Threat Research | Oct 18 2018 |

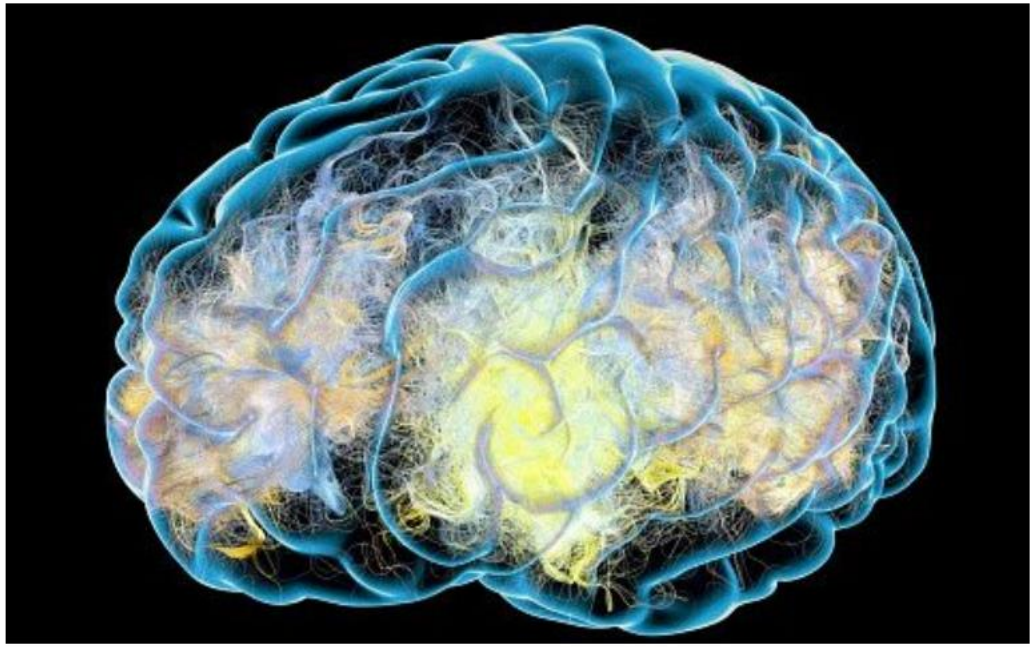
Researchers: Ori Karliner (@oriHCX)

Relevant Operating Systems: FreeRTOS up to V10.0.1 (with FreeRTOS+TCP), AWS FreeRTOS up to V1.3.1, WHIS OpenRTOS and SafeRTOS (With WHIS Connect middleware TCP/IP components) .

CVE List:

CVE	Description
CVE-2018-16522	Remote code execution

Brain implants used to treat Parkinson's can be hacked and used to control people, scientists warn



Hackers could overload or stop brain implants remotely, scientists have claimed CREDIT: NAEBLYS

By Natasha Bernal
31 OCTOBER 2018 • 11:33AM

Brain implants used to treat Parkinson's disease could be hacked by cyber attackers and used to control people, scientists have claimed.

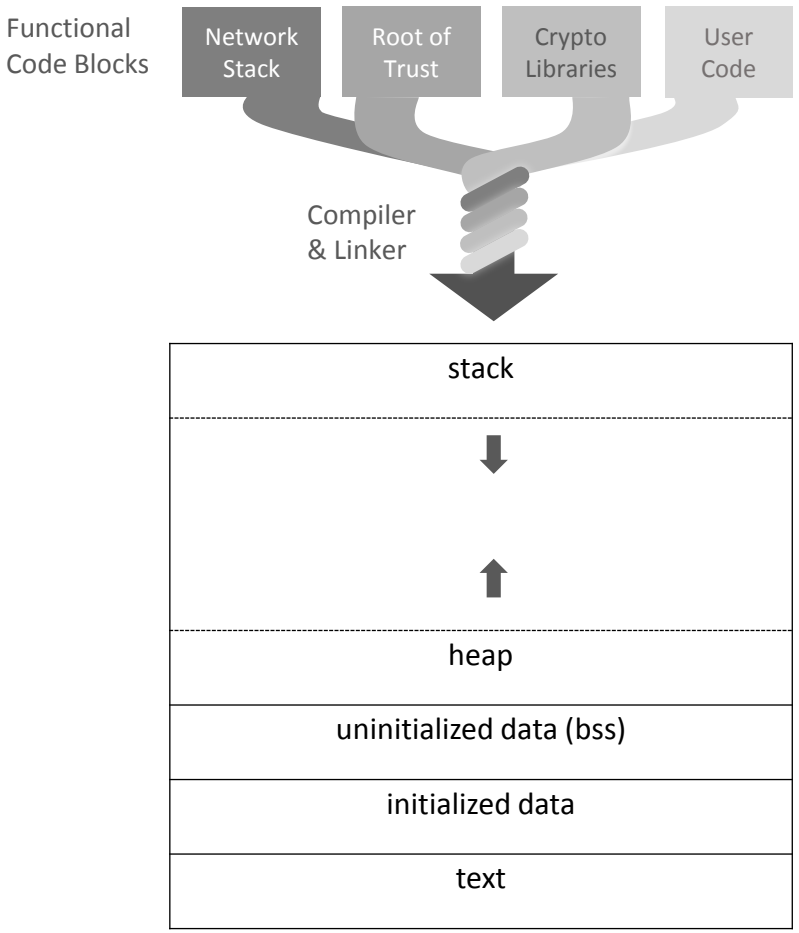
HEX-Five™

Making RISC-V the Most Secure Platform Ever

Cesare Garlati, Hex Five Security

RISC-V Summit – December 4, 2018

Traditional computing is not safe nor secure



All programs share the same access to code and data

- Compiler & linker combine code blocks into monolithic binaries
- Functional blocks communicate via shared memory – stack, heap, buffers
- Any bit of code can access any bit of data – can't hide secrets
- One faulty / malicious instruction can stall / crash the whole system – DOS
- MMUs that run 17M lines of "rich OS" can't be trusted either – see enclaves

System engineering – uncomfortable commercial reality

- No company can afford the cost and the time to develop a full sw stack in house
- Open source software, 3rd party libraries, "untouchable" legacy code are the norm
- The design complexity of security technologies often results in them not being used

Virtual Memory (MMU) Uncomfortable Truth

```
~/linux-4.18.6$ cloc --exclude-lang=DTD,Lua,make .
   60965 text files.
   60546 unique files.
   14391 files ignored.

-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                            25782        2554166        2248398       12965944
C/C++ Header                 18693         484773         892818        3629746
Assembly                      1318          47155         105960         232515
JSON                           189              0              0         102201
Perl                            55           5414           3994          27294
Bourne Shell                   346           5633           4983          24450
Python                         108           3055           3337          17427
HTML                             5            669              0           5492
yacc                             9            701            375           4648
lex                              8            326            314           2007
C++                              7            285             77           1844
Bourne Again Shell              51            351            318           1711
awk                             11            170            155           1384
Markdown                         1            220              0           1077
TeX                              1            108              3            915
NAnt script                      2            156              0            599
Windows Module Definition        2             14              0            102
m4                               1             15              1             95
XSLT                             5             13             26             61
CSS                              1             18             27             44
vim script                       1              3             12             27
Ruby                             1              4              0             25
INI                              1              1              0              6
sed                              1              2              5              5
-------------------------------------------------------------------------------
SUM:                         46599        3103252              0       17019619
-------------------------------------------------------------------------------
```

(a) Industry Average: "about 15 - 50 errors per 1000 lines of delivered code." He further says this is usually representative of code that has some level of structured programming behind it, but probably includes a mix of coding techniques.

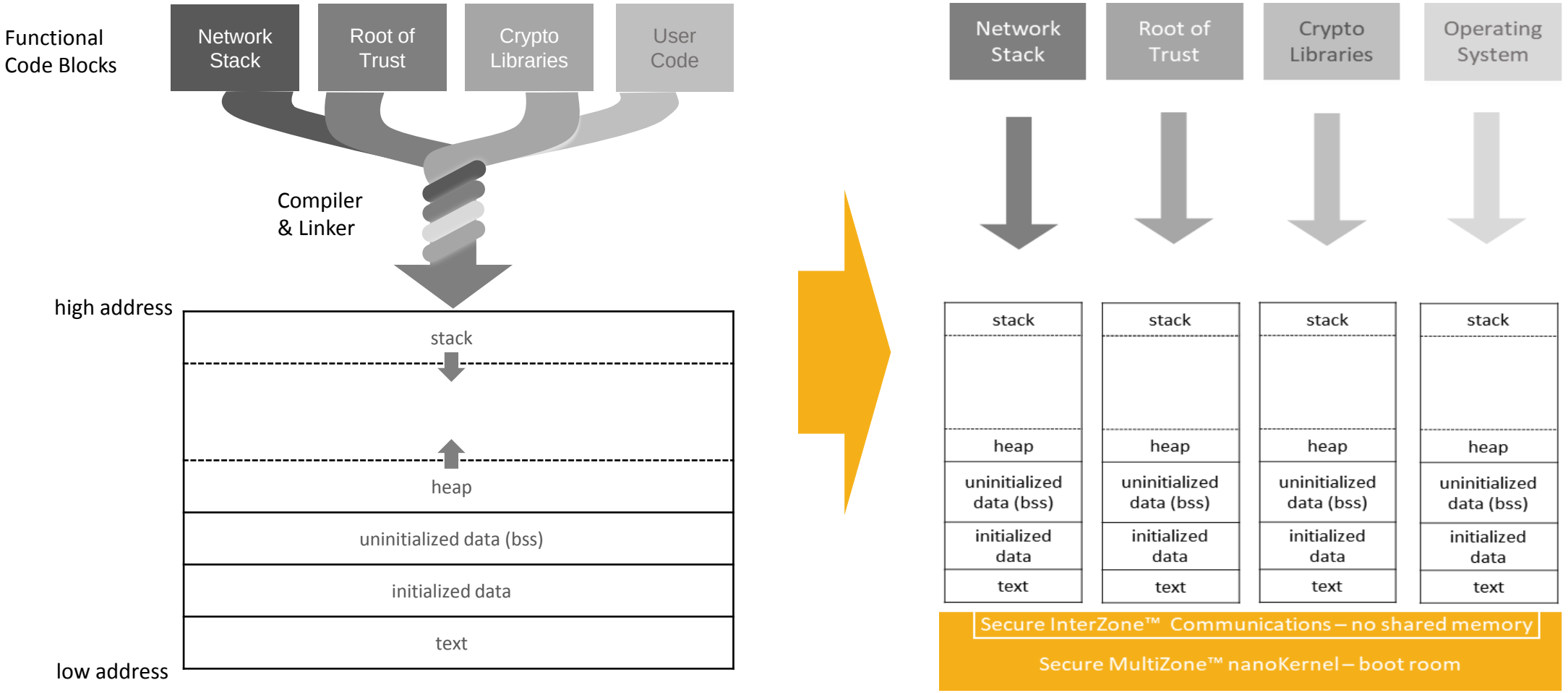
(b) Microsoft Applications: "about 10 - 20 defects per 1000 lines of code during in-house testing, and 0.5 defect per KLOC (KLOC IS CALLED AS 1000 lines of code) in released product (Moore 1992)." He attributes this to a combination of code-reading techniques and independent testing (discussed further in another chapter of his book).

(c) "Harlan Mills pioneered 'cleanroom development', a technique that has been able to achieve rates as low as 3 defects per 1000 lines of code during in-house testing and 0.1 defect per 1000 lines of code in released product (Cobb and Mills 1990).

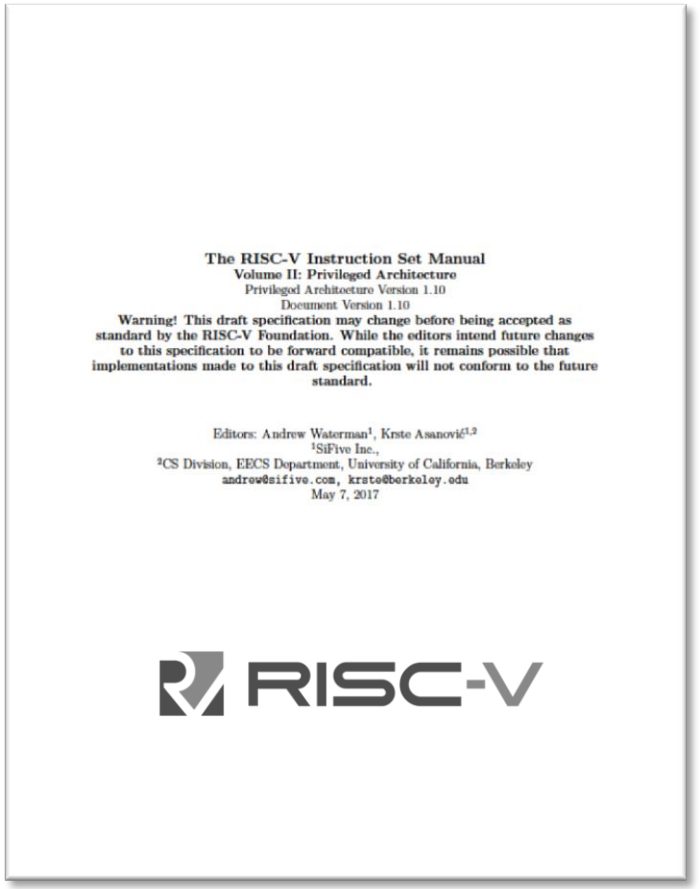
17,019,619 * 10^{-4} = **1,701** disasters waiting to happen

Credits: Al Danial https://github.com/AlDanial/cloc, Dan Mayer's development blog https://www.mayerdan.com/ruby/2012/11/11/bugs-per-line-of-code-ratio

A Better Way – Multidomain Trusted Execution Environment



RISC-V Security 101



The RISC-V Instruction Set Manual
Volume II: Privileged Architecture
Privileged Architecture Version 1.10
Document Version 1.10
Warning! This draft specification may change before being accepted as standard by the RISC-V Foundation. While the editors intend future changes to this specification to be forward compatible, it remains possible that implementations made to this draft specification will not conform to the future standard.

Editors: Andrew Waterman[1], Krste Asanović[1,2]
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
May 7, 2017

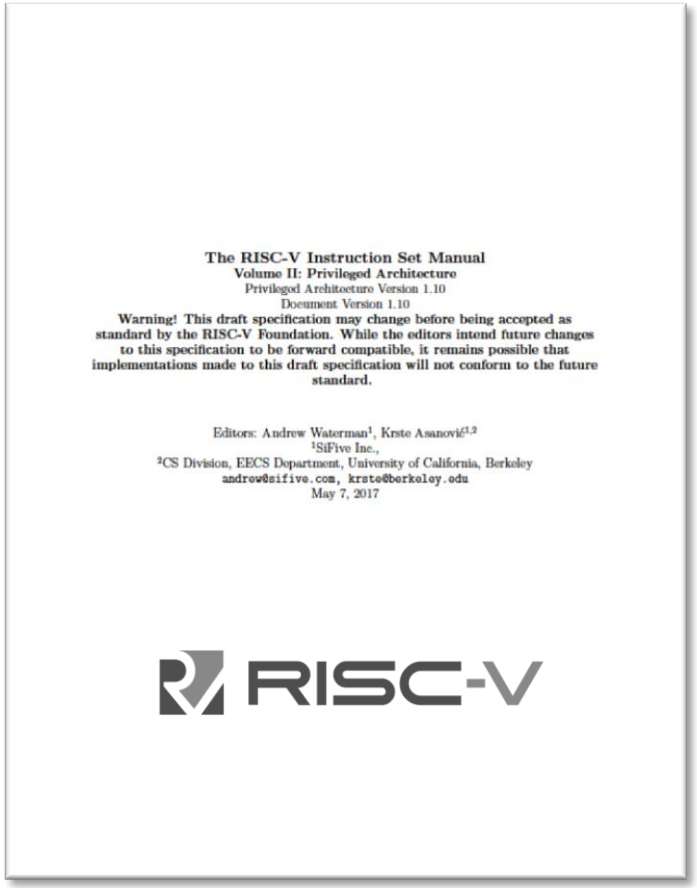
RISC-V

Privilege Levels (rings) – Control and Status Registers (CSRs)

- Machine – always present
- Supervisor – Linux
- Reserved (Hypervisor) – work in progress
- User / Application – i.e. SiFive E21/E31/E51, Andes N25 / NX25
- Trusted Execution Environment runs at highest privilege level
- Note: Interrupts always M mode (unless "N" implemented)

Number of levels	Supported Modes	Intended Usage
1	M	Simple embedded systems
2	M,U	Secure embedded systems
3	M,S,U	Unix-like operating systems

RISC-V Security 101



The RISC-V Instruction Set Manual
Volume II: Privileged Architecture
Privileged Architecture Version 1.10
Document Version 1.10
Warning! This draft specification may change before being accepted as standard by the RISC-V Foundation. While the editors intend future changes to this specification to be forward compatible, it remains possible that implementations made to this draft specification will not conform to the future standard.

Editors: Andrew Waterman[1], Krste Asanović[1,2]
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
May 7, 2017

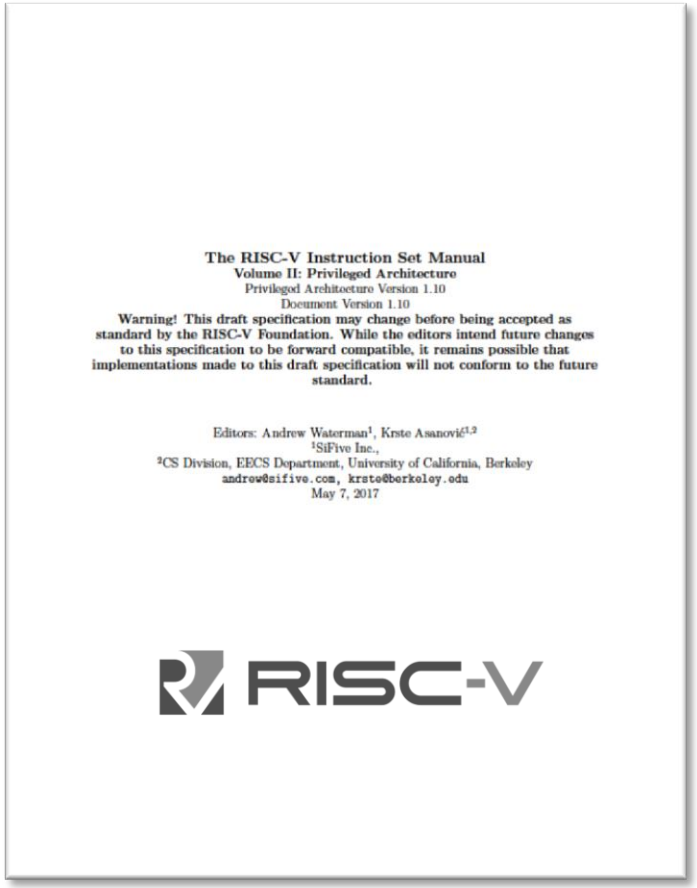
RISC-V

Physical Memory Attibutes / Physical Memory Protection

- Hardware enforced – 4 ranges * 4 config reg (if implemented)
- Policy R/W/X => synchronous exception mechanism (trap)
- Overlapping OK, ranges can be locked down
- Top of range (TOR) or naturally aligned power of two (NAPOT)
- Trusted Execution Environment manages PMP context at runtime
- Note: enforced per core – no ISA spec for multi-core / platform

A	Name	Description
0	OFF	Null region (disabled)
1	TOR	Top of range
2	NA4	Naturally aligned four-byte region
3	NAPOT	Naturally aligned power-of-two region, ≥8 bytes

RISC-V Security 101



The RISC-V Instruction Set Manual
Volume II: Privileged Architecture
Privileged Architecture Version 1.10
Document Version 1.10
Warning! This draft specification may change before being accepted as standard by the RISC-V Foundation. While the editors intend future changes to this specification to be forward compatible, it remains possible that implementations made to this draft specification will not conform to the future standard.

Editors: Andrew Waterman[1], Krste Asanović[1,2]
[1]SiFive Inc.,
[2]CS Division, EECS Department, University of California, Berkeley
andrew@sifive.com, krste@berkeley.edu
May 7, 2017

RISC-V

User-mode Interrupts and exception handling ("N")

- Outer ring delegates designated interrupts to user-level
- Hardware transfers control directly to U-mode – but no security ctx
- Intended for secure embedded systems with only M + U
- Optional extension – no known commercial implementations yet
- Trusted Execution Environment provides "N" in sw via trap & emulate

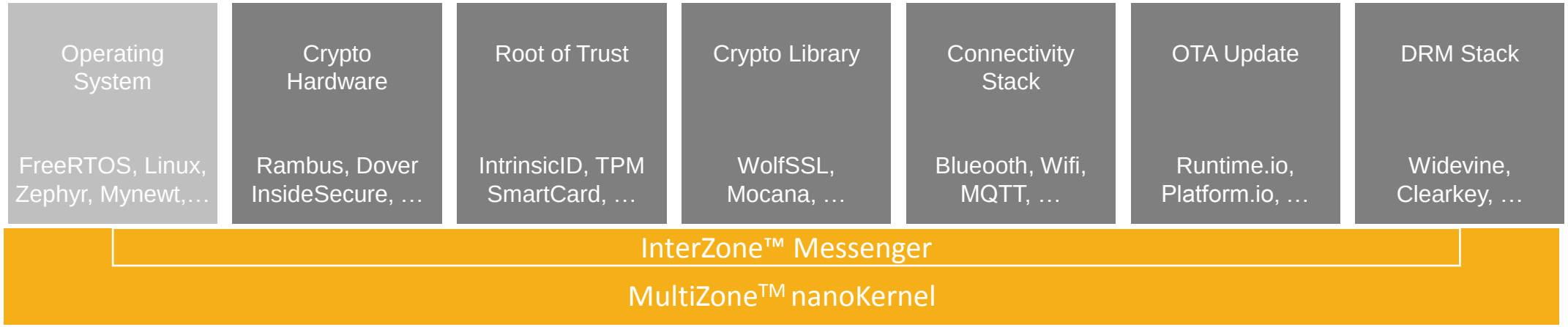
Number	Name	Description
0x000	ustatus	User status register.
0x004	uie	User interrupt-enable register.
0x005	utvec	User trap handler base address.
0x040	uscratch	Scratch register for user trap handlers.
0x041	uepc	User exception program counter.
0x042	ucause	User trap cause.
0x043	utval	User bad address or instruction.
0x044	uip	User interrupt pending.

How all it comes together: MultiZone™ Trusted Execution Environment

"MultiZone it's like Docker at the chip level – just way better"

- Unlimited number of equally secure zones – ram, rom, i/o
- Hardware-enforced, Software-enabled, Policy-driven RWX
- Supports fencing and user mode interrupts – even without "N"
- Formally verifiable nanoKernel completely written in assembly
- Consumes <1% of cpu and <1KB of ram, fits every boot rom
- Does not require changes to existing code – elf / hex sources
- Built from the ground up for RISC-V – no clunky legacy porting
- Completely based on RISC-V standard extensions
- Free and open source API under Apache v2 available on GitHub

MultiZone™ Trusted Execution Environment



Enable ***proper*** implementation of all typical security components for RISC-V systems

- OOB pre-certified implementations – wrap traditional APIs with InterZone™ Messages
- Open standards interoperable solution – OEMs focus on differentiation
- Low friction path to market – Hex Five in every RISC-V boot ROM, SDK and custom design IP

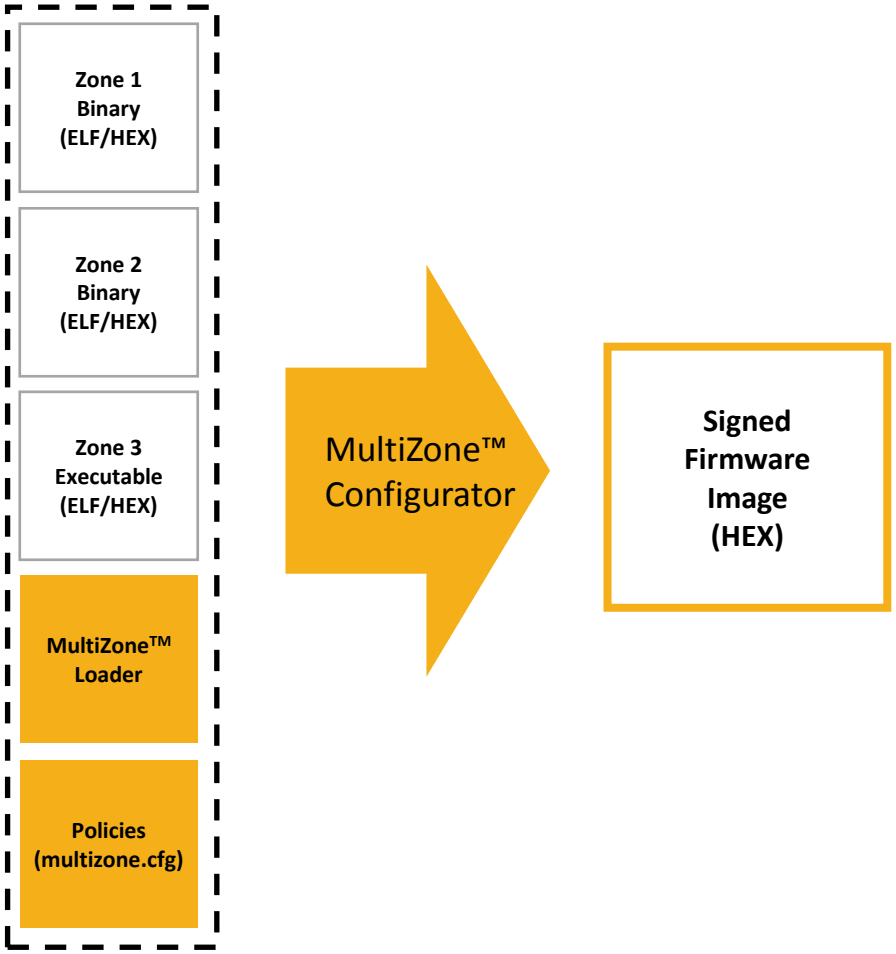
*"the design complexity associated with properly implementing [these] security technologies often results in them not being used at all" - **Brandon Lewis**, Editor-in-Chief, Embedded Computing Design*

MultiZone™ – How It Works

Open | multizone.cfg ~/eclipse-cdt-ws/hexfive-conf | Save

```
# Copyright(C) 2018 Hex Five Security, Inc. - All Rights Reserved

# Kernel
tick  = 10      # ms
rtc   = 32768   # Hz
sched = PREEMPT # PREEMPT | COOP

# Zone 1
mz1_fence = FENCE
mz11_base = 0x40410000; mz11_size = 64K; mz11_rwx = RX # FLASH
mz12_base = 0x80001000; mz12_size =  4K; mz12_rwx = RW # RAM
mz13_base = 0x20000000; mz13_size =  32; mz13_rwx = RW # UART

# Zone 2
mz2_irq = 11, 21, 22 # BTN0 BTN1 BTN2
mz21_base = 0x40420000; mz21_size = 64K; mz21_rwx = RX # FLASH
mz22_base = 0x80002000; mz22_size =  4K; mz22_rwx = RW # RAM
mz23_base = 0x0200BFF8; mz23_size =   8; mz23_rwx = RO # RTC
mz24_base = 0x20005000; mz24_size =  64; mz24_rwx = RW # PWM
mz25_base = 0x20002000; mz25_size =  64; mz25_rwx = RW # GPIO
mz26_base = 0x0C000000; mz26_size =  4M; mz26_rwx = RW # PLIC

# Zone 3
mz3_irq = 23 # BTN3
mz31_base = 0x40430000; mz31_size = 64K; mz31_rwx = RX # FLASH
mz32_base = 0x80003000; mz32_size =  4K; mz32_rwx = RW # RAM
mz33_base = 0x0200BFF8; mz33_size =   8; mz33_rwx = RO # RTC
mz34_base = 0x20002000; mz34_size =  64; mz34_rwx = RW # GPIO
```

Plain Text | Tab Width: 3 | Ln 10, Col 1 | INS



MultiZone™ Free and Open API

```
/* Copyright 2018 Hex Five Security, Inc.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or impl
See the License for the specific language governing permissions and
limitations under the License. */

void ECALL_YIELD();
void ECALL_SEND(int, void *);
void ECALL_RECV(int, void *);

void ECALL_TRP_VECT(int, void *);
void ECALL_IRQ_VECT(int, void *);

uint64_t ECALL_CSRR_MTIME();
uint64_t ECALL_CSRR_MCYCLE();
uint64_t ECALL_CSRR_MINSTR();
uint64_t ECALL_CSRR_MHPMC3();
uint64_t ECALL_CSRR_MHPMC4();

uint64_t ECALL_CSRR_MISA();
uint64_t ECALL_CSRR_MVENDID();
uint64_t ECALL_CSRR_MARCHID();
uint64_t ECALL_CSRR_MIMPID();
uint64_t ECALL_CSRR_MHARTID();
```

```
void button_0_handler(void) __attribute__((interrupt("user")));
void button_0_handler(void){ // global interrupt

plic_source int_num  = PLIC_claim_interrupt(&g_plic); // claim

LED1_GRN_ON; LED1_RED_OFF; LED1_BLU_OFF;

GPIO_REG(GPIO_RISE_IP) |= (1<<BUTTON_0_OFFSET); //clear gpio irq

PLIC_complete_interrupt(&g_plic, int_num); // complete

ECALL_SEND(1, ((int[]){201,0,0,0}));

}

/*configures Button0 as a global gpio irq*/
void b0_irq_init()  {

  //disable hw io function
  GPIO_REG(GPIO_IOF_EN )    &=  ~(1 << BUTTON_0_OFFSET);

  //set to input
  GPIO_REG(GPIO_INPUT_EN)   |= (1<<BUTTON_0_OFFSET);
  GPIO_REG(GPIO_PULLUP_EN)  |= (1<<BUTTON_0_OFFSET);

  //set to interrupt on rising edge
  GPIO_REG(GPIO_RISE_IE)    |= (1<<BUTTON_0_OFFSET);

  ECALL_IRQ_VECT(11, button_0_handler);

}
```

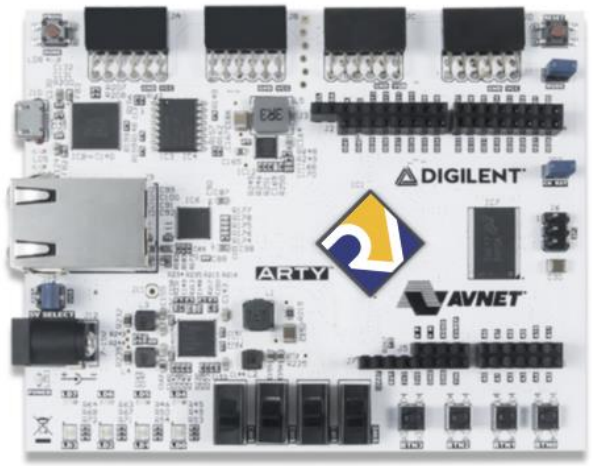
MultiZone™ Reference Application

Zone 1	Zone 2	Zone 3
Terminal Console	irq demo app	real time ctrl app
Ilm + dlm +uart	Ilm + dlm + plic + gpio	Ilm + dlm + spi + gpio + rtc)

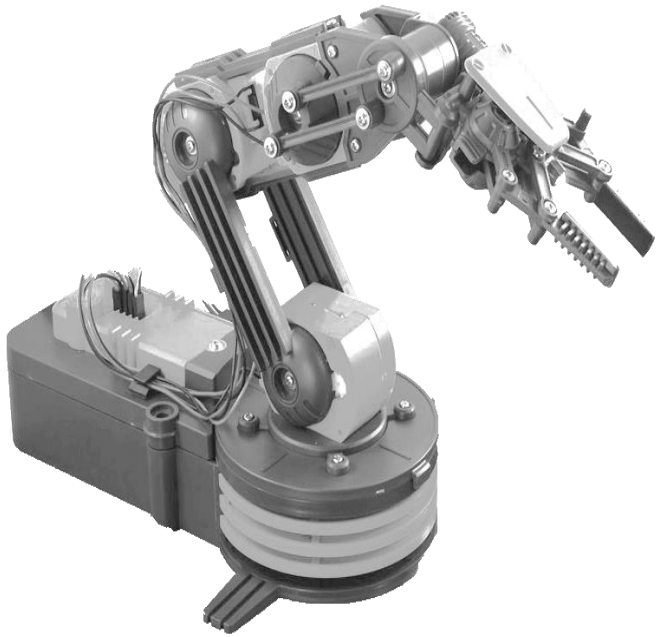
InterZone™ Secure Communications

MultiZone™ nanoKernel

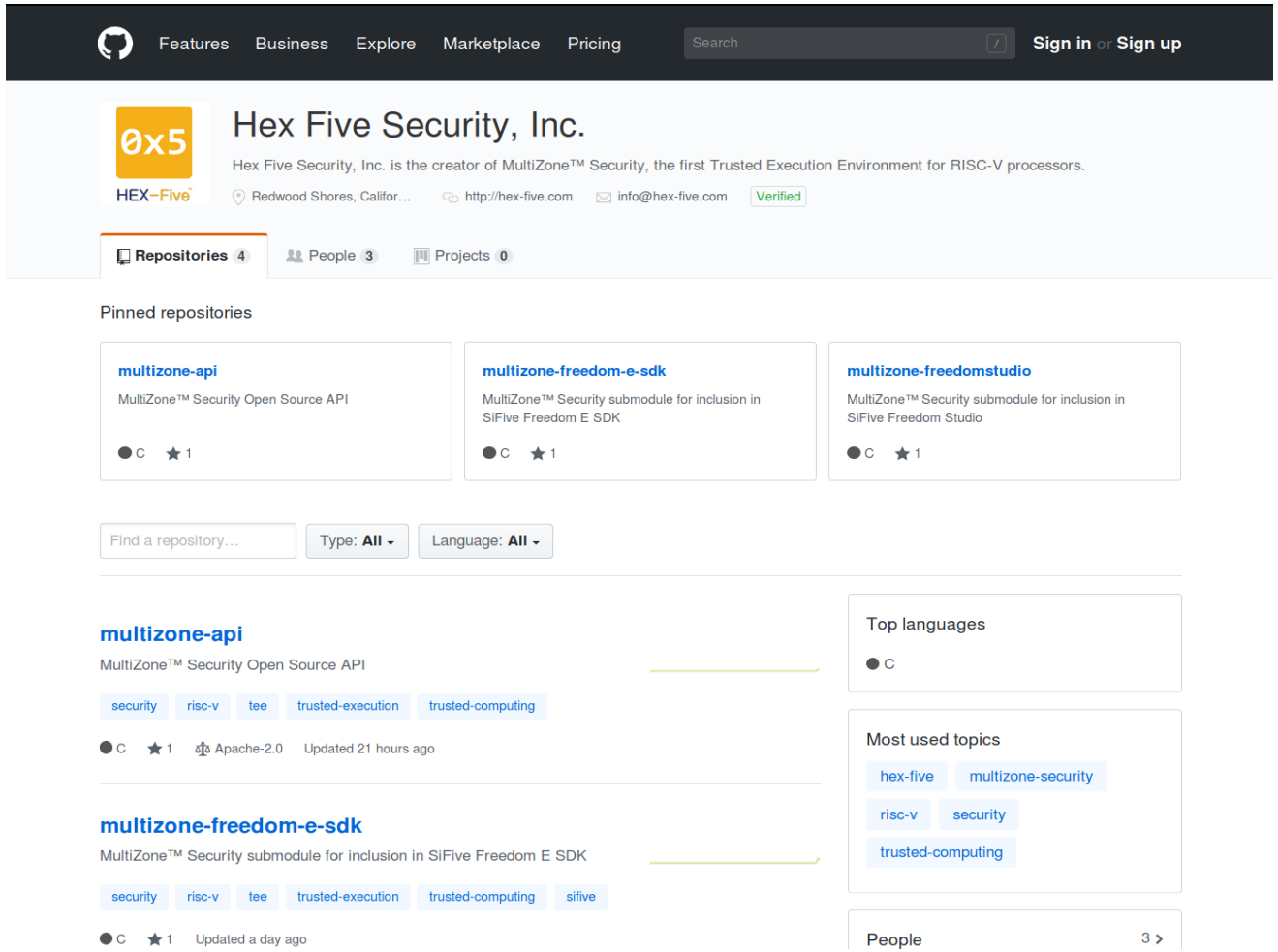
UART



SPI / USB



Open Source License – Apache v2



Hex-Five multizone api is licensed under the Apache License 2.0

A permissive license whose main conditions require preservation of copyright and license notices. Contributors provide an express grant of patent rights. Licensed works, modifications, and larger works may be distributed under different terms and without source code.

Commercial & Free Eval Licenses

Serverless | M³ | CLL | Events | Whitepapers | The Next Platform

The Register®
Biting the hand that feeds IT

Security

Arms race: SiFive, Hex Five build code safe houses for RISC-V chips

Those developing custom CPUs can now tap a TrustZone-ish trusted execution environment

By Thomas Claburn in San Francisco 10 Sep 2018 at 20:08 2

GlobeNewswire

ANDES TECHNOLOGY

Hex Five adds MultiZone Security to the Andes RISC-V Cores on GOWIN FPGAs

The first RISC-V Trusted Execution Environment Enabling Robust Security for Andes N(X)25 RISC-V Cores on the GOWIN GW-2A Family of FPGAs

November 09, 2018 11:00 ET | **Source:** Andes Technology Corporation

San Jose, CA , Nov. 09, 2018 (GLOBE NEWSWIRE) -- Hex Five Security, Inc, the creator of MultiZone™ Security, Andes Technology Corporation and GOWIN Semiconductor Corp announce a collaboration to enable MultiZone™ Security, the first Trusted Execution Environment for RISC-V on the Andes N(X)25 RISC-V Cores, which is part of 25-series, with the GOWIN GW-2A Family of FPGAs.

Hex Five's patent pending technology provides policy-based hardware-enforced separation for an unlimited number of security zones, with full control over data, code, interrupts and peripherals. MultiZone's™ Configurator takes fully compiled and linked customer code and merges it with Hex Five's nanoKernel to enable rapid adoption without any changes to hardware or customer code basis.

Andes 32-bit N25(F)/A25, and 64-bit NX25(F)/AX25 are versatile CPU cores compliant to RISC-V ISA that deliver over 3.5

Takeaways

- Embedded systems – with or without MMU – are inherently not secure as all code can access all data and peripherals.
- The RISC-V ISA defines some security building blocks including privileged modes, physical memory protection and user-mode interrupts.
- However, the design complexity associated with properly implementing these technologies often results in them not being used at all.

1. *MultiZone™ Security provides an unlimited number of equally secure execution environments.*
2. *MultiZone™ Security provides hardware-enforced policy-defined separation for code, data and I/O.*
3. *MultiZone™ Security doesn't require additional cores, specialized IP or changes to existing applications.*

0x5

HEX–Five™

Hex Five MultiZone™

Hex Five Security, Inc. is the creator of MultiZone™, the first Trusted Execution Environment for RISC-V. Hex Five patent pending technology provides policy-based hardware-enforced separation for an unlimited number of security domains, with full control over data, code and peripherals. Contrary to traditional solutions, Hex Five MultiZone™ requires no additional cores, specialized hardware or changes to existing software. Open source libraries, third party binaries and legacy code can be configured in minutes to achieve unprecedented levels of safety and security.

Hex Five - MultiZone™ Security

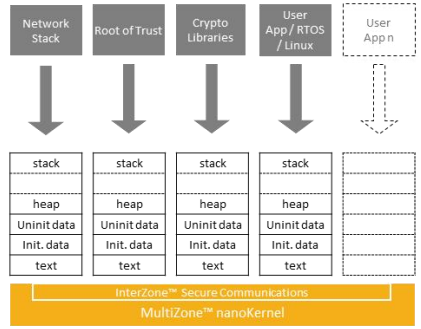
Hex Five Security, Inc. is the creator of MultiZone™ Security, the first Trusted Execution Environment (TEE) for RISC-V. Hex Five's patent pending technology provides policy-based hardware-enforced separation for an unlimited number of security domains, with full control over data, code and peripherals.

What is MultiZone™ Security?

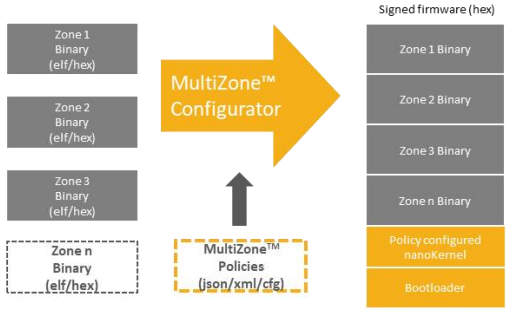
MultiZone™ Security is the first Trusted Execution Environment for RISC-V – it enables development of a simple, policy-based security environment for RISC-V that supports rich operating systems through to bare metal code.

MultiZone™ Security consists of the following components:

- ***MultiZone™ nanoKernel*** – lightweight, formally verifiable, bare metal kernel providing policy-driven hardware-enforced separation of ram, rom, i/o and interrupts.
- ***InterZone™ Messenger*** – communications infrastructure to exchange secure messages across zones on a no- shared memory basis.
- **MultiZone™ Configurator** – combines fully linked zone executables with policies and kernel to generate the signed firmware image.
- **MultiZone™ Signed Boot** – 2-stage signed boot loader to verify integrity and authenticity of the firmware image (sha-256 / ECC)



How does MultiZone™ Security Integrate with Existing Development Environment?



MultiZone™ Security integrates seamlessly into your existing IDE such as Eclipse or command line based toolset.

Application blocks are written, compiled and linked separately for each zone producing a set of elf or hex file.

MultiZone™ Policies are set to achieve the desired ram, rom, i/o and interrupt isolation for each zone – RWX, with granularity down to 4 bytes.

Finally the MultiZone™ configurator is invoked to combine zone elf/hex files with the nanoKernel and bootloader into a signed firmware image.

The full system can be written, compiled and debugged with your existing GNU or Eclipse toolset.

www.hex-five.com

info@hex-five.com

Hex Five - MultiZone™ Security

Hex Five Security, Inc. is the creator of MultiZone™ Security, the first trusted execution environment (TEE) for RISC-V. Hex Five's patent pending technology provides policy-based hardware-enforced separation for an unlimited number of security domains, with full control over data, code and peripherals.

Features

- Preemptive real time scheduler: round robin / cooperative, configurable time tick, cpu overhead < 1%
- Formally verifiable, completely written in assembly, self-contained - no 3rd party library dependencies
- Unlimited number of isolated Trusted Execution Environments (zones) - hardware-enforced, policy-defined
- Up to 32 memory-mapped resources per zone – i.e. flash, ram, i/o, uart, gpio, timers, etc.
- Any combination of top-of-range and naturally aligned configuration – minimum granularity 4 bytes
- Any combination of read, write, execute policy – resource overlapping allowed although not recommended
- Built-in support for fencing configurable on a per-zone basis – i.e. cache / pipeline / instruction / load /store
- Full support for PLIC and CLIC Interrupts – fully configurable zone / interrupt mapping
- Full support for secure user-mode interrupt handlers – even without 'N' extensions
- Full support for low-latency vectored interrupts, preemptable interrupts, and Wait For Interrupt - suspend mode
- Built in trap & emulate for most protected instructions – i.e. CSR read only
- Secure inter-zone communications infrastructure based on messaging - no shared memory / buffers
- C library wrapper for protected mode execution – via ECALL exception handling mechanism
- Signed boot suitable for 2-stage boot room and/or public key / root of trust / PUF – SHA-256 / ECC
- Command line configurator utility compatible with any operating system capable of running Java 1.8

Development Environments

- Eclipse IDE including MCU and GNU Toolchain plugins and OpenOCD / JTAG / GDB live debugging
- SiFive FreedomStudio IDE including MCU and GNU Toolchain plugins and OpenOCD / JTAG / GDB live debugging
- SiFive Freedom E SDK – command line based
- Linux and Windows command line tools (make, gcc, gdb, etc.) – native Linux, Java 1.8 required for Windows
- Built-in Board Support Packages for SiFive E31 and E21 bitstreams (Xilinx ARTY FPGA)

System Requirements

- 32 bit or 64 bit RISC-V ISA with 'S' or 'U' extensions
- Physical Memory Protection compliant with Ver. 1.10
- 4KB FLASH and 1KB RAM

Hex Five Security is a proud member of the RISC-V Foundation

0x5 HEX-Five Security RISC-V

www.hex-five.com

info@hex-five.com