RISC-V Multi Zone Enclave Free And Open Standard API

Cesare Garlati, Hex Five Security

OSEW 2019
Evolution of Hardware Security

No Security

Arm® TrustZone®

RISC-V Multi Zone Trusted Execution Environment

Arm and TrustZone are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.
### RISC-V Multi Zone Secure IoT Stack

<table>
<thead>
<tr>
<th>Zone #1</th>
<th>Zone #2</th>
<th>Zone #3</th>
<th>Zone #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO / IRQs</td>
<td>ETHERNET</td>
<td>OTP / FUSE / PUF</td>
<td>UART</td>
</tr>
</tbody>
</table>

**Internet**

**Crypto**
- TLSv1.3, Cipher TLS_AES_128_GCM_SHA256
- Peer signing digest: SHA256
- Peer signature type: ECDSA
- Server Temp Key: ECDH, P-256, 256 bits
- Server public key is 256 bit
- Private Key ASN1 OID: prime256v1
- Private Key NIST CURVE: P-256

**U-mode Apps**

**U-mode IRQs**

**M-mode**

**PMP Hardware**

**ARGET FPGA - Rocket RV32 IMACU**

**0x5 HEX-Five Security**
RISC-V Multi Zone Enclave – SMP Linux

Hardware-enforced Software-defined Boundaries (PMP)

- Enclave #1: Terminal (bare metal), IPC Driver
- Enclave #2: Terminal (bare metal), IPC Driver
- Enclave #3: UART FPGA Board, PWM (LED)
- Enclave #4: TEE Console (bare metal), IPC Driver

MultiZone™ Secure Communications
MultiZone™ nanoKernel
Hex Five E51 (RV64 IMACU)

- Unlimited number of statically defined enclaves – ram, rom, i/o, irq
- Secure messaging with no shared mem – secure buffers for SMP Linux
- Secure interrupt handlers mapped to enclaves and executed in U-mode
- Trap & emulation of privileged instructions, Soft-timers, Secure boot
- Free and open std API maintained by Hex Five – ISC Permissive License
RISC-V Multi Zone Enclave – Data Model

| Zone 1 | Binary (ELF/HEX) |
| Zone 2 | Binary (ELF/HEX) |
| Zone 3 | Binary (ELF/HEX) |
| Zone 4 | Binary (ELF/HEX) |

nanoKernel

Policies

multizone.cfg

Tick = 10 # ms

Zone 1
base = 0x08000000; size = 32K; rwx = rx; # FLASH
base = 0x08001000; size = 4K; rwx = rw; # RAM

Zone 2
base = 0x08010000; size = 32K; rwx = rx; # FLASH
base = 0x08002000; size = 4K; rwx = rw; # RAM

Zone 3
base = 0x08018000; size = 32K; rwx = rx; # FLASH
base = 0x08003000; size = 4K; rwx = rw; # RAM
base = 0x2000104000; size = 0x100; rwx = rw # UART0 EXP

Zone 4
base = 0x20040000; size = 64K; rwx = rw # FLASH
base = 0x08004000; size = 4K; rwx = rw # RAM
base = 0x10020000; size = 0x100; rwx = rw # PWM

Hart = 1, 2, 3, 4
base = 0x01000000; size = 256; rwx = rw # DTIM0 [0x0100_0000 - 0x0100_00FF] 256
base = 0x01000000; size = 8K; rwx = --- # DTIM0 [0x0100_0000 - 0x0100_1FFF] 8K
base = 0x01800000; size = 8K; rwx = --- # ITIM0 [0x0180_0000 - 0x0180_1FFF] 8K
base = 0x08000000; size = 128K; rwx = --- # L2LIM [0x0800_0000 - 0x0801_FFFF] 128K
base = 0x20000000; size = 8M; rwx = --- # FLASH [0x2000_0000 - 0x207F_FFFF] 8M
base = 0x20000400; size = 8; rwx = --- # MTIMECMP0
base = 0x10020000; size = 0x100; rwx = --- # PWM
base = 0x00000000; size = 256G; rwx = rwx # ACCESS ALL [0x0 - 0x3F_FFFF_FFFF] 256G
RISC-V Multi Zone Enclave – API

/* Copyright(C) 2019 Hex Five Security, Inc.
Permission to use, copy, modify, and/or distribute this software for
any purpose with or without fee is hereby granted, provided that the
above copyright notice and this permission notice appear in all copies.
*/

#ifndef LIBHEXFIVE_H_
define LIBHEXFIVE_H_

void ECALL_YIELD();
void ECALL_WFI();
int ECALL_SEND(int, void *);
int ECALL_RECV(int, void *);
void ECALL_TRP_VECT(int, void *);
void ECALL_IRQ_VECT(int, void *);
void ECALL_CSR5_MIE();
void ECALL_CSR5_MIE();
void ECALL_CSRW_MTIMECMP(uint64_t);

uint64_t ECALL_CSR5_MTME();
uint64_t ECALL_CSR5_MCYCLE();
uint64_t ECALL_CSR5_MINSTR();
uint64_t ECALL_CSR5_MHPMC3();
uint64_t ECALL_CSR5_MHPMC4();
uint64_t ECALL_CSR5_MISA();
uint64_t ECALL_CSR5_MVENDOR();
uint64_t ECALL_CSR5_MARCHID();
uint64_t ECALL_CSR5_MIMPID();
uint64_t ECALL_CSR5_MAHRID();

#endif /* LIBHEXFIVE_H_ */

Permissive Licensing – “any purpose”

Hardware threads (zones) management
Inter zone messaging – zone0 SMP Linux
Traps & IRQs handlers registration (U-mode)
Traps & IRQs enable / disable – per zone
Hardware thread timer – per zone

Trap & emulation helpers
Read-only, selected CSRs
Completely optional – just for speed / latency
/* 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);
}

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);
    }
Hex Five MultiZone™ Security

Hex Five Security, Inc. is the creator of MultiZone™ Security, the first trusted execution environment for RISC-V. Hex Five open standard 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™ Security 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.

http://hex-five.com