RISC-V Enclaves
A Clean Slate Approach To Linux Security

Cesare Garlati, CEO Hex Five Security

RISC-V Summit 2019
Multiple trusted workloads on untrusted platforms – i.e. rich OS

Linux / RTOS / Legacy OS

Sharing the same core(s) is not secure: M-mode SBI & L1/L2

Zero-trust: assume the supply chain is compromised – i.e. drivers

Zero-tolerance for thick hypervisors and large codebase sys software
(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
RISC-V Security Primitives

Privilege Levels & Control and Status Registers

- **Machine** – always present, highest privilege mode
- **Supervisor** – Linux, supports MMU / virtual memory
- **Reserved (Hypervisor)** – work in progress
- **User / Application** – unprivileged lowest level
- **Trusted Execution Environment** runs at highest privilege
- **Note:** Interrupts always M mode (unless “N” implemented)

<table>
<thead>
<tr>
<th>Rings</th>
<th>Modes</th>
<th>Intended Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>M</td>
<td>Unsecured embedded</td>
</tr>
<tr>
<td>2</td>
<td>M,U</td>
<td>Secure embedded</td>
</tr>
<tr>
<td>3</td>
<td>M,S,U</td>
<td>Linux</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TOR</td>
<td>Top of range</td>
</tr>
<tr>
<td>NA4</td>
<td>Naturally aligned 4-byte</td>
</tr>
<tr>
<td>NAPOT</td>
<td>Naturally aligned power of 2</td>
</tr>
</tbody>
</table>
MultiZone™ Security For Linux

Multiple equally secure zones – ram, rom, i/o, irq handlers
Hardware-enforced, Software-defined, Policy-driven (rwx)
Extremely lightweight: codebase < 2KB, formally verifiable

The only commercial Linux enclave for RISC-V
Easy to configure and deploy – toolchain extension
No need to rewrite software: runs unmodified binaries
MultiZone™ Security – How It Works

MultiZone™ Security PolarFire Edition – Live Demo

SSH – Enclave #1
SSH – Enclave #2
UART – Enclave #3
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 software-defined hardware-enforced separation for multiple security domains, with full isolation of data, programs and peripherals. Contrary to traditional solutions, MultiZone™ Security requires no additional 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.
MultiZone™ Open Standard API – C Library

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

/* 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_CSRM_MIE();
void ECALL_CSRM_MIE();

void ECALL_CSRW_MTIMECMP(uint64_t);

uint64_t ECALL_CSRM_MTIME();
uint64_t ECALL_CSRM_MCYCLE();
uint64_t ECALL_CSRM_MINSTR();
uint64_t ECALL_CSRM_MPMC3();
uint64_t ECALL_CSRM_MPMC4();

uint64_t ECALL_CSRM_MISC();
uint64_t ECALL_CSRM_MVENDORID();
uint64_t ECALL_CSRM_MARCHID();
uint64_t ECALL_CSRM_MIMPID();
uint64_t ECALL_CSRM_MHARTID();

#endif /* LIBHEXFIVE_H */