Zen Crypted Operating System

Технічне завдання

на розробку та впровадження операційної системи OS.1 на гіпервізорі seL4

На основі Технічних Вимог

Згідно з Постановою КМУ № 205 від 21.02.2025

Формалізованих за стандартом ISO/IEC/IEEE 29148:2018

OS.1: High-Assurance BEAM on seL4 with Alpine VMM for drivers

Namdak Tonpa (maxim@synrc.com), Synrc Research

10 серпня 2026 р.

Зміст

Анотація

The Erlang/OTP BEAM runtime provides robust concurrency, fault isolation, and soft real-time behaviour that make it attractive for high-availability systems, yet it traditionally depends on a large general-purpose operating system whose trusted computing base (TCB) undermines strong security and certification arguments. We present Synrc Hypervision, a hybrid architecture that hosts an unmodified BEAM on the formally verified seL4 microkernel while retaining practical device support through a minimal Alpine Linux guest.

A purpose-built thin host (Synrc) implements only the small set of Linux-compatible system calls required by a musl-linked OTP runtime and executes as an isolated seL4 protection domain under the Microkit framework. Device drivers that would otherwise enlarge the TCB are confined to an Alpine Linux virtual machine managed by a lightweight virtual-machine monitor; communication between the BEAM domain and the driver domain occurs exclusively through capability-mediated shared-memory channels defined by the seL4 Device Driver Framework (sDDF). The resulting system therefore combines the formal isolation guarantees of seL4, the minimal attack surface of a BEAM-specific host, and the hardware coverage of an existing Linux driver ecosystem.

We describe the static system architecture, the mapping of selected NIST SP 800-53 control families onto seL4 capabilities and compile-time configuration, and a concrete bootstrap path on commodity hardware (Raspberry Pi 4). The design demonstrates that a production-grade BEAM application can operate with a dramatically reduced TCB while preserving the ability to utilise complex devices, offering a practical route toward higher-assurance, certifiable distributed systems.

1. Introduction

Erlang/OTP and its BEAM virtual machine have long been valued for building highly concurrent, fault-tolerant distributed systems. The language-level process model, supervision trees, and soft real-time scheduling provide strong application-level isolation and recovery. However, these benefits sit atop a conventional general-purpose operating system (typically Linux) whose large trusted computing base (TCB) remains a significant obstacle to high-assurance and certification arguments.

Formal isolation at the kernel level has advanced substantially with the seL4 microkernel, which offers a machine-checked proof of functional correctness and integrity. Yet running a full Linux guest under seL4 merely relocates the TCB problem. Conversely, pure unikernel approaches that eliminate the general-purpose OS often force substantial rewriting of application runtimes or sacrifice device-driver coverage.

Synrc Hypervision addresses this tension by combining three elements:

- the formally verified seL4 microkernel and its Microkit framework for static, capability-based system construction;
- Synrc, a minimal host that supplies only the approximately fifty Linux-compatible system calls required by a musl-linked, unmodified OTP BEAM;
- a carefully confined Alpine Linux virtual machine that supplies real device drivers, communicating with the BEAM domain solely through seL4 Device Driver Framework (sDDF) shared-memory channels.

The resulting architecture keeps the TCB under the BEAM extremely small (seL4 + Synrc) while still permitting practical use of complex hardware. All isolation boundaries, scheduling parameters, and device ownership are fixed at image-construction time, yielding a static system amenable to analysis and aligned with selected families of NIST SP 800-53 controls.

This paper presents the architecture in detail, describes the principal protection domains and their interactions, and outlines the compile-time configuration model that governs resource allocation and scheduling.

2. Architecture

Synrc Hypervision is organised as a static collection of seL4 protection domains (PDs) whose memory regions, communication channels, interrupt bindings and scheduling contexts are declared in a Microkit system description and materialised at build time. No protection domain or channel can be created dynamically at runtime.

2.1. Components and Links

The principal components and the morphisms that connect them are summarised below.

Component	Type	Responsibility
seL4	Microkernel	Isolation, capabilities, MCS scheduling, IRQ routing
Microkit	Framework	Static system description → concrete capability layout
Synrc	Protection Domain	Syscall trap surface and minimal host for BEAM
BEAM / OTP	Application runtime	Erlang/Elixir processes, schedulers, supervision
VMM (libvmm)	Protection Domain	Creates and manages the Alpine Linux guest
Alpine Linux guest	Virtual Machine	Real device drivers (network, block, . . .)
UIO helper	User-space (Linux)	Bridges Linux drivers ↔ sDDF rings
sDDF	Protocol + libraries	Shared-memory rings and notifications
Console PD	Protection Domain	Early serial / console output
Monitor PD	Protection Domain	Metrics, health, audit events
Storage PD	Protection Domain	Block / SSD access (native or via Alpine)
Network PD	Protection Domain	Ethernet access (native or via Alpine)

The dominant data and control paths are:

- **Network**: NIC → Alpine driver → UIO helper → sDDF ring → Synrc → BEAM `gen_tcp/Bandit`;
- **Storage**: SSD/block device → Storage path → sDDF → Synrc → BEAM file or ETS/Mnesia;
- **Monitoring**: Synrc, VMM and driver PDs → notification + shared metrics region → Monitor PD;
- **Console**: UART/virtio-console owned exclusively by the Console PD exposed to Synrc.

All authority is expressed by seL4 capabilities granted at system construction. There is no ambient authority.

2.2. seL4 Microkit Capabilities

Microkit provides a deliberately constrained component model on top of seL4:

- **Protection Domains (PDs)** — isolated, single-threaded components with fixed address spaces (maximum 63 PDs in a system);
- **Memory Regions** — explicitly mapped with read/write/execute rights into one or more PDs;
- **Channels** — point-to-point connections supporting asynchronous notifications and synchronous protected procedure calls;
- **Virtual Machines** — a PD may own at most one guest VM;
- **MCS scheduling contexts** — each PD receives a budget and period; core affinity is declared statically.

The system description is processed at build time. The resulting image contains a fixed capability layout; the Microkit initialiser installs these capabilities before any PD begins execution. Consequently, the set of resources each component may access is known a priori and can be audited against security policy (including selected NIST SP 800-53 families such as AC, SC and CM).

Scheduling follows the seL4 Mixed-Criticality Scheduling (MCS) model. Typical priority ordering (highest to lowest) is:

1. Console PD,
2. Synrc PD (hosting BEAM),
3. Monitor PD,
4. VMM / Alpine driver VM,
5. optional background domains.

Core affinity and budget/period parameters are supplied via compile-time flags (`CORES`, `AFFINITY`, etc.) and emitted into the system description.

2.3. VMM Protection Domain

The VMM protection domain encapsulates the Linux driver environment so that complex device support does not enlarge the TCB under BEAM.

2.3.1. libvmm

libvmm is the lightweight virtual-machine monitor library used by Microkit. It runs as an ordinary protection domain and is responsible for:

- allocating guest physical memory,
- establishing stage-2 translation,
- injecting interrupts,
- mediating hypercalls, and
- granting the guest only those device registers and IRQs that policy permits.

The VMM itself holds only the capabilities necessary to manage its single guest; it cannot map arbitrary host memory or interfere with Synrc or other native PDs.

2.3.2. Alpine

Alpine Linux (musl-based, minimal) is chosen as the guest operating system for three reasons: small footprint, binary compatibility with the musl-linked OTP that Synrc expects, and straightforward packaging. A Buildroot- or minirootfs-derived image contains:

- a stripped kernel configured for the target board and for UIO,
- a minimal user-space (busybox, openrc or equivalent),
- the UIO helper binary that maps sDDF rings and translates between Linux driver interfaces and the sDDF protocol.

Only the devices assigned to the guest appear in its device tree; all other hardware remains invisible.

2.3.3. Synrc

Synrc executes as a sibling protection domain to the VMM, not inside the Linux guest. It implements a deliberately narrow Linux-compatible system-call surface (approximately fifty calls) that a musl-linked OTP binary expects: memory management (`mmap`, `mprotect`, ...), threading and synchronisation (clone, futex family), time, basic file descriptors, and the in-memory VFS that holds the OTP release.

Synrc never owns real device registers. All I/O is performed by mapping sDDF rings that are filled by either native driver PDs or by the UIO helper running inside Alpine. Consequently the trusted computing base that sits directly under BEAM consists solely of seL4 and Synrc.

2.3.4. BEAM

An unmodified (or minimally patched) musl-linked OTP release is embedded as a cpio or tar archive inside the Synrc protection domain. At start-up Synrc locates the archive, constructs the necessary environment, and transfers control to `beam.smp`. From that point the ordinary OTP boot sequence proceeds: schedulers are started, applications are loaded, and supervision trees become active. BEAM continues to manage its own processes, message passing and distribution; seL4 only guarantees the CPU budget and memory isolation of the enclosing Synrc domain.

2.4. Monitor Protection Domain

The Monitor PD collects health and performance data from Synrc, the VMM and any native driver domains. Communication occurs through a combination of seL4 notifications and a shared metrics memory region whose layout is fixed at build time. The Monitor may:

- expose counters and timestamps to an external observer,
- generate audit-style events (supporting NIST AU-family controls),
- detect liveness failures and signal higher-level recovery logic.

When the compile-time flag `MONITOR=0` is supplied the entire domain is omitted, further reducing image size and TCB.

2.5. Console Protection Domain

The Console PD owns the platform UART (or a virtio-console device) exclusively. It provides a narrow, character-oriented interface to Synrc for early boot messages, panic output and optional interactive debugging. Because the device registers reside only in the Console PD's capability space, neither Synrc nor the Alpine guest can access them directly. The domain can be compiled to a write-only sink for production images.

2.6. Storage Protection Domain

Storage may be realised in two ways, selected at image-construction time:

- **Native sDDF block driver** — a dedicated PD owns the block device registers and IRQ and speaks the sDDF block protocol to Synrc;
- **Alpine-mediated** — the block device is passed through to the Linux guest; the UIO helper translates Linux block I/O into sDDF rings.

In both cases Synrc presents a simple block or file-system abstraction to BEAM. The choice is expressed by the `STORAGE` build flag and the corresponding entries in the Microkit system description. Device ownership is exclusive: a given controller is never mapped into more than one domain.

2.7. Network Protection Domain

Analogously to storage, networking is provided either by a native sDDF Ethernet driver PD or by the Alpine guest. The data path is identical in structure:

$$\text{NIC} \rightarrow (\text{native PD or Alpine} + \text{UIO}) \rightarrow \text{sDDF ring} \rightarrow \text{Synrc} \rightarrow \text{BEAM socket layer.}$$

Compile-time selection (`NET=1`) determines which implementation is linked into the final image. Core affinity for the network domain can be set independently so that packet processing does not perturb BEAM's soft real-time behaviour.

2.8. NIST SP 800-53 Control Mapping

Synrc Hypervision's static, capability-based design maps naturally onto several families of NIST SP 800-53 controls. The table below summarises the primary alignments. This is a design-time correspondence, not a formal certification claim.

Family	Example Controls	How Synrc Hypervision addresses them
AC Access Control	AC-3, AC-4, AC-6	seL4 capabilities and Microkit channels enforce least privilege and information-flow control; no ambient authority exists at runtime.
AU Audit & Accountability	AU-2, AU-12	Monitor PD collects events and metrics over explicit notification channels; audit data can be exported or retained in a dedicated region.
CM Configuration Management	CM-2, CM-6, CM-7	All isolation boundaries, device ownership and feature sets are fixed at image-construction time; unused components are compiled out.
SC System & Communications Protection	SC-7, SC-39	Strong spatial isolation between PDs; boundary protection is realised by capability checks rather than software firewalls.
SI System & Information Integrity	SI-7, SI-16	Static, reproducible images; capability-based memory protection; optional measured-boot hooks.
CP Contingency Planning	CP-9, CP-10	BEAM supervision trees provide deterministic process-level recovery inside the Synrc domain; seL4 contains failures to the originating PD.
IA Identification & Authentication	IA-2, IA-5	Optional; can be realised as a dedicated protection domain that authenticates external principals before granting channels.
SA System & Services Acquisition	SA-11, SA-15	Small TCB, open design, reproducible builds, and clear separation of upstream components (seL4, Microkit, Synrc, Alpine).

The compile-time flags (`SECURITY=nist53`, `MONITOR=1`, `CORES`, `AFFINITY`, etc.) directly influence the strength of the above controls by determining which domains are present, which devices are accessible, and how CPU time is partitioned.

3. Conclusion

Synrc Hypervision demonstrates that a production-grade, unmodified BEAM runtime can be hosted on the formally verified seL4 microkernel with a dramatically reduced trusted computing base while still retaining practical access to complex devices. The architecture achieves this by:

- confining the BEAM to a minimal syscall-trap host (Synrc) that runs as an ordinary seL4 protection domain;
- isolating device drivers inside a capability-restricted Alpine Linux virtual machine (or native sDDF PDs);
- mediating all inter-component communication through static, capability-controlled sDDF channels;
- fixing scheduling parameters, core affinity and device ownership at image-construction time.

The resulting system is static, analysable, and naturally aligned with key NIST SP 800-53 control families. Future work includes completing native sDDF drivers for the Raspberry Pi platform, strengthening measurement and attestation support, and evaluating end-to-end performance and fault-isolation properties under realistic OTP workloads.