

Kernel Builds Deserve an Identity: Portable Kernel Artifacts and Traceable Test Results

Cong Wang, Founder and CEO, Multikernel Technologies, Inc.

Linux Plumbers Conference 2026

Thousands of Kernels a Day, None with a Name

"Boot failure on arm64, 6.12.0-rc3, defconfig, lab X". Which binary? Can I boot it?

Identifier	What it misses
<code>uname -r</code>	A string: <code>LOCALVERSION</code> makes it anything
Git commit	Source, not config, toolchain, or binary
CI build ID	Assigned by each CI, not derived from the binary

Userspace got OCI: content-addressed, typed, pulled anywhere. The kernel never did.

KBI in One Slide

The kernel as its own OCI image, versioned independently of userspace:

```
/kbi/vmlinuz  initrd  modules/<kver>/  firmware/  symvers  
  
kbi_id = "kbi:sha256:" + sha256(sort([  
    "vmlinuz:" + sha256(vmlinuz), "btf:" + ..., "config:" + ... ]))
```

- Computed, not assigned: same inputs, same ID, on any machine, no registry involved
- Each artifact is a typed layer; no change to the OCI spec
- **Add-ons bind to an ID**: mismatches are refused before boot

Where It Stands

- Go CLI, Apache 2.0: build, push, pull, install, sign, verify
- About 5,800 lines including tests; cosign-compatible signatures with threshold policies
- Runs on bare metal (install, then your bootloader) and in VM or image builds

Honest gaps:

- No measured boot; resolve does not re-check module contents
- No deep BPF verification: kfunc signatures, CO-RE relocations
- **No CI system emits KBI images today. That is why we are here**

Q1: Where Is the Line for the ID?

Today the ID hashes `vmlinux`, BTF, and config. Modules, initrd, and firmware are left out.

- The same `vmlinux` with and without BTF gets two IDs
- Bundled modules are checked against `--kver`, not hashed
- A firmware refresh does not change the kernel's identity

Should BTF and config be in? Should modules? What is the one question the ID answers?

Q2: Artifact Identity or Recipe Identity?

The ID hashes the binary. Two builds match only if the build is reproducible.

- **Artifact identity**: what booted. Verifiable, fragile across rebuilds
- **Recipe identity**: tree + config + toolchain. Stable, unverifiable from the binary
- **Both**, with the recipe recorded as metadata on the artifact

Which one does a regression report need?

Can we ask CI for reproducible builds?

Q3: Could CI Emit Bundles?

- **Tuxmake**: output a KBI image alongside today's artifacts
- **KernelCI Maestro**: push bundles to a registry
- **KCIDB**: add a verifiable build identity field

Then a regression report becomes a `kbi pull`, not a description.

What blocks each of these? Who hosts the registry, and for how long?

Q4: How Far Toward a Kernel SBOM?

Traceability needs more than "which binary":

- **Source:** commit, patches, tree
- **Build:** config, toolchain, flags
- **Contents:** modules, firmware licenses, BPF objects

Where should it stop? SPDX, CycloneDX, or both?

Inside the OCI image, or beside it?

Q5: Who Signs a CI Kernel?

`kbi verify` supports threshold policies, so a kernel can require more than one signature.

- **The CI** that built it: proves where it came from
- **The maintainer** whose tree it is: proves it is wanted
- **The lab** that tested it: proves it booted

Which of these matter downstream?

Without measured boot, what does a signature buy?

Q6: Where Should Pack Verification Live?

Today KBI checks modules against `Module.symvers`. BPF packs declare the kfuncs they need.

- **In KBI**: reimplement what the loader and verifier check
- **In libbpf and kmod**: expose an offline "would this load?" mode
- **In the kernel build**: emit the metadata a checker needs

Is an offline check worth having, and who should own it?

Q7: Bare Metal CI with Multikernel

Today: reboot, flash, wait. With multikernel:

- Host kernel stays up and runs the lab
- `kerf` spawns a test kernel, pulled by KBI ID, on spare cores
- It crashes alone; results are tagged with its KBI ID
- Many test kernels, one machine, in parallel

Not wired up yet. **Is this the lab you want? What would it fail to catch?**

Further Discussion

Contact: cwang@multikernel.io

Code: github.com/multikernel/kbi

Mailing list: multikernel@lists.linux.dev