SEC-LINE OS (zencrypted/os)

Technical Whitepaper: A Monolithic, Zero-Trust Architecture for Hardened Edge and Desktop Systems

SEC-LINE Engineering Group / bitedits

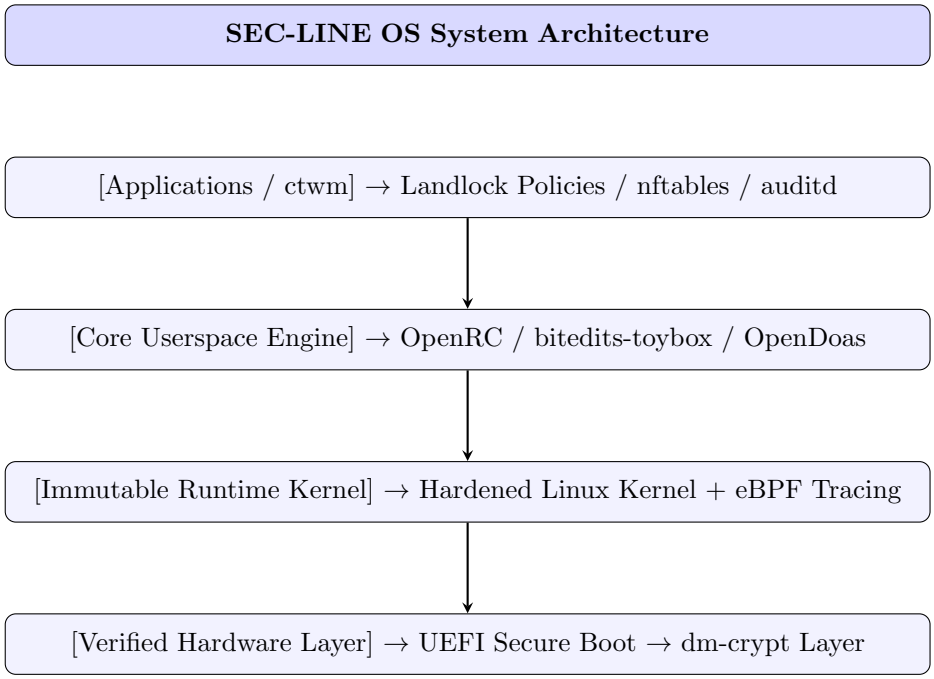
Status: Architecture Specification — Classification: Public
July 28, 2026

Contents

1 Executive Summary

Modern operating systems suffer from bloated attack surfaces, architectural coupling with system management daemons (`systemd`), and complex boot chains vulnerable to physical and logical tampering. **SEC-LINE OS** (internally designated as **zencrypted/os**) introduces a minimalist, zero-trust operating infrastructure designed for high-assurance computing.

By synthesizing an ultra-lightweight independent rootfs framework, an immutable Unified Kernel Image (UKI), declarative hardware sandboxing, and strict privilege isolation, SEC-LINE OS eliminates legacy attack vectors. The operating system boots directly from UEFI NVRAM into an encrypted memory-mapped environment, bypassing traditional bootloaders, and executing a sandboxed userspace governed by `musl-libc`, `toybox`, and the Landlock Linux Security Module (LSM).



2 Architectural Pillars

The design of SEC-LINE OS relies on four uncompromising principles:

- **Cryptographic Immutability:** The entire base operating system kernel, configuration matrix, and initial execution files reside in a single, cryptographically signed binary.

- **Attack Surface Minimization:** Traditional distributions deploy up to several gigabytes of software. SEC-LINE OS strips the base rootfs down to its absolute mathematical minimum via custom forks (`bitedits/*`).
- **Process Isolation by Default:** Applications lack system-wide visibility. Access to the filesystem, network socket layers, and hardware inputs must be explicitly granted via declarative kernel policies.
- **State Separation:** The operating system treats runtime state as volatile. Persistent changes are restricted to dedicated, encrypted data blocks.

3 The Secure Boot & Storage Cryptography Pipeline

SEC-LINE OS completely eliminates secondary bootloaders (e.g., GRUB, systemd-boot), which represent significant structural targets for Evil Maid attacks and persistent firmware exploitation.

3.1 Unified Kernel Image (UKI) Construction

The system builds a monolithic `zencrypted-os.efi` file containing:

1. An official UEFI boot stub (`systemd-stub` stripped of systemd execution logic).
2. A highly tailored, hardened Linux kernel.
3. The system kernel command line (hardcoded at compile-time to prevent boot parameter injection).
4. A minimal, specialized stage-1 `initramfs`.

3.2 Cryptographic Verification

During machine activation, the hardware firmware evaluates the UKI against custom Platform Keys (`PK`/`KEK`/`db`) enrolled in the motherboard NVRAM. If verification succeeds, control hands off directly to the kernel execution thread.

3.3 Early-Stage Disk Decryption (`dm-crypt`)

The immutable `initramfs` instantiates a minimal storage driver stack to invoke Linux kernel cryptography routines.

- The primary system partition is validated and decrypted using `cryptsetup` with `dm-crypt` (AES-XTS-PLAIN64).
- Storage headers are detached or randomized to obscure partition bounds.
- Once authentication passes, the root filesystem mounts as a read-only device, executing a `switch_root` loop into the main operating workspace.

4 Kernel Architecture & Security Profiles

The operating system runs an optimized downstream derivation of the Linux Hardened kernel, prioritizing compile-time exploit mitigation over raw benchmark optimizations.

4.1 Core Hardening Flags

The kernel configuration enforces:

- Strict Page Table Isolation (`PAGE_TABLE_ISOLATION=y`).
- Kernel Stack Randomization on system calls (`RANDOMIZE_KSTACK_OFFSET=y`).
- Complete memory wiping on allocation/deallocation boundary conditions (`INIT_ON_ALLOC_DEFAULT_ON=y`).

4.2 Mandatory Access Control via Landlock LSM

Unlike legacy MAC systems (SELinux/AppArmor) which add complex policy layers and complex tooling, SEC-LINE OS implements process sandboxing via **Landlock**.

- **Mechanism:** Utilizing unprivileged sandboxing features, processes restrict their own filesystem actions and access paths.
- **Configuration:** The userspace engine utilizes `landlockconfig` to ingest declarative TOML rule structures at application initialization.

Example System Profile: /etc/landlock/ctwm.toml

```
[sandbox]
handled_access_fs = ["execute", "read_file", "read_dir", "write_file"]

[[rules.fs]]
path = "/usr/bin/ctwm"
access = ["execute", "read_file"]

[[rules.fs]]
```

```
path = "/home/zencrypted/.ctwmrc"
access = ["read_file"]
```

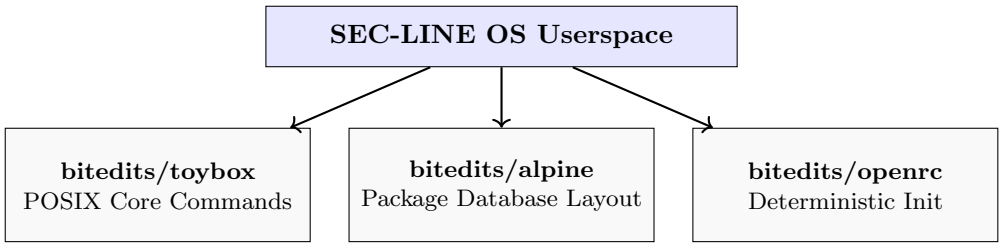
4.3 High-Fidelity Tracing & Auditing (eBPF & auditd)

System behavior analysis relies on non-invasive inspection loops rather than intrusive hooking infrastructures.

- **Auditing:** System call profiling is verified by `bitedits/audit-userspace`, capturing unauthorized privilege transition vectors and immediate file write signals.
- **eBPF Observability:** Leveraging `bpftrace` patched directly for `musl-libc` interfaces, security daemons monitor kernel tracepoints dynamically without requiring run-time structural alterations.

5 Userspace Composition & Fork Strategy

SEC-LINE OS avoids standard GNU utils and `busybox` monoliths, selecting modular components customized and maintained under the `bitedits` repository tree.



5.1 POSIX Engine: `bitedits/toybox`

All vital system tools (`ls`, `ps`, `cat`, `chmod`) are provided via Rob Landley's `toybox`. This deployment yields:

- A clean breakpoint separating the system from bloated GNU software dependencies.
- Strict compliance with POSIX runtime directives.
- Symmetrical performance enhancements when interacting with memory operations over the `musl` API.

5.2 Service Supervision: `bitedits/openrc`

Process initialization follows a deterministic dependency graph managed by `openrc`.

- **No Parallel Background Bloat:** Service dependencies map cleanly during build orchestration.
- **Predictable Execution Blocks:** Services run as unprivileged isolations where feasible, logging securely to stateless `/dev/log` pipes audited by `auditd`.

5.3 Privilege Escalation: OpenDoas

For administrative access loops, standard `sudo` is banned. The system relies on `bitedits/OpenDoas`, enforcing:

- A code footprint less than 5% of legavcy `sudo`.
- Zero context persistence between separate shell instances.
- Strict execution constraints mapped explicitly per user profile.

6 Networking, Peripheral & Graphical Environments

The user interaction domain operates with the same minimal privileges as the core data routing architecture.

6.1 Network Security Layer (`nftables`)

The firewall stack loads directly within the kernel image definition, adopting a strict **Default-Drop** posture across all interface hooks:

nftables Core Configurations

```
table inet filter {chain input {type filter hook input priority filter;
policy drop;ct state established,related acceptiif "lo" accept}chain
↪  forward {type filter hook forward
priority filter; policy drop;}chain output {type filter hook output
↪  priority filter; policy drop;
udp dport 123 accept # Allowed: chrony NTP Synctcp dport 443 accept #
↪  Allowed: Outbound Encrypted TLS}}
```

6.2 Seat and Input Management (`seatd`)

To operate a graphical workspace without a resource-heavy or complex display manager, SEC-LINE OS links input routing through `seatd`.

- Grants application engines access to shared hardware interfaces (such as graphics cards or inputs) without requiring elevated `root` permissions.
- Avoids any dependency on large user tracking packages, maintaining a tiny security perimeter for input tracking.

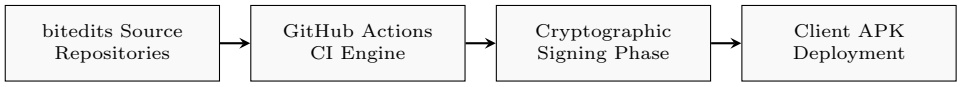
6.3 Graphical Topography (`ctwm`)

The desktop layout is anchored by `ctwm` (Claude's Tab Window Manager).

- **Efficiency:** It runs inside an extremely lightweight system profile, bypassing modern desktop notification loops and complex message broker backplanes.
- **Security:** Minimizes the visual display and rendering attack surface against local interface sniffing vectors.

7 Package Lifecycle & Supply Chain Integrity

Software supply chains represent a primary target for sophisticated adversaries. SEC-LINE OS maintains integrity through a strict binary deployment loop.



1. **Isolated Source Compilation:** Custom `APKBUILD` files fetch package source trees directly from validated `bitedits/*` endpoints.
2. **Deterministic Compilation:** Packages compile within automated, isolated pipelines generating reproducible hash check validations.
3. **Cryptographic Asset Signing:** Package outputs are packaged using Alpine's native Package Keeper (`apk-tools`) format and signed with an offline master key pair.
4. **Local Repository Integration:** The rootfs image builder installs signed `.apk` payloads directly into the OS structure, refusing any package lacking verified, local cryptographic signatures.

8 Comparative Matrix

Table 1: System Parameter Architectural Baseline

Security Parameter	Standard Enterprise Linux	SEC-LINE OS
Boot Mechanism	UEFI → GRUB → Kernel → Initramfs	UEFI → Signed Monolithic UKI
Boot Surface Tampering	Vulnerable (Unsigned configs/modules)	Impossible (Breaks Secure Boot Signature)
Core Utilities	GNU Coreutils (∼150MB, Complex)	`bitedits/toybox` (<5MB, Minimalist)
Init Framework	Complex asynchronous initialization	Monolithic, simple OpenRC system
Default Sandbox Engine	Process-wide (SELinux/AppArmor)	Thread-level declarative Landlock
Privilege Transition	Legacy Sudo Engine (Highly complex)	OpenDoas Framework (Audited, Minimal)

9 Conclusion

SEC-LINE OS proves that modern hardware systems can be decoupled from complex, legacy distribution frameworks. By consolidating the runtime layer inside an immutable Unified Kernel Image and enforcing strict sandboxing across minimal userspace forks, SEC-LINE OS delivers an uncompromised, zero-trust infrastructure optimized for high-assurance desktop layouts and isolated edge computation nodes.