PhantomOS — Complete Technical Document

Version: 1.0 **Architecture**: x86_64 bare-metal **Philosophy**: "To Create, Not To Destroy"

Table of Contents

---

1. Overview

PhantomOS is a bare-metal x86_64 operating system written entirely from scratch in C and x86_64 assembly. It runs directly on hardware (or in QEMU) with no dependency on any existing OS, standard library, or runtime. The kernel boots via Multiboot2/GRUB, transitions from 32-bit protected mode to 64-bit long mode, sets up paging, and initializes a complete graphical desktop environment with 25 interactive applications.

The core philosophy — **"To Create, Not To Destroy"** — is enforced at every level. The filesystem is append-only (data is hidden, never deleted). The AI Governor monitors all

operations and transforms destructive requests into safe alternatives. Every policy decision is audited. Statistics never decrease.

Key Numbers

Metric	Value
Total kernel source lines	~36,500
Desktop GUI alone	9,502 lines
Interactive applications	25
Shell commands	31
Governor policies	13
Framebuffer depth	32 bits per pixel
Default resolution	1024×768
Max concurrent windows	32
GeoFS volume size	1MB content + 256KB refs + 128KB views

---

2. Boot Process

Stage 1: Multiboot2 Entry (32-bit)

The kernel image begins with a Multiboot2 header (magic `0xE85250D6`) requesting a 1024×768×32bpp framebuffer. GRUB loads the kernel at physical address 1MB (`0x100000`) and passes a multiboot info structure.

On entry at `_start`: 1. Verify multiboot magic number 2. Copy multiboot info to a safe buffer (GRUB may place it where the stack will be) 3. Set up a 32-bit stack at `0x14f000` 4. Verify CPU supports CPUID and Long Mode via `cpuid`

Stage 2: Long Mode Transition

Page tables are constructed to identity-map the first 1GB using 2MB huge pages:

Structure	Address	Purpose
PML4	`0x148000`	Page Map Level 4 (1 entry)
PDPT	`0x149000`	Page Directory Pointer Table (1 entry)
PD	`0x14a000`	Page Directory (512 × 2MB entries = 1GB)
Stack	`0x14f000`	Kernel stack top

The transition sequence: 1. Load CR3 with PML4 physical address 2. Enable PAE in CR4 (bit 5) 3. Set Long Mode Enable in EFER MSR (`0xC0000080`, bit 8) 4. Enable paging in CR0 (bits 0 + 31) 5. Far jump to 64-bit code segment (`0x08`)

Stage 3: 64-bit Kernel Entry

After entering long mode: 1. Reload segment registers with 64-bit data selector (`0x10`) 2. Set 64-bit stack pointer 3. Call `kmain(multiboot_info_ptr, multiboot_magic)`

GDT Layout

Selector	Type	Descriptor
`0x00`	Null	Required null descriptor
`0x08`	Code	64-bit code segment (`0x00209A0000000000`)
`0x10`	Data	64-bit data segment (`0x0000920000000000`)

Linker Layout (at 1MB)

```
0x100000  .multiboot2   Multiboot2 header (must be in first 32KB)
          .text         Kernel code (page-aligned)
          .rodata       Read-only data (page-aligned)
          .data         Initialized data (page-aligned)
          .bss          Uninitialized data (page-aligned)
```

Symbols exported: `__kernel_start` , `__kernel_end` , `__bss_start` , `__bss_end`

---

3. Memory Management

Physical Memory Manager (PMM)

Bitmap-based allocator tracking 4KB pages. Initialized from the Multiboot2 memory map.

Function	Description
`pmm_alloc_page()`	Allocate single 4KB page
`pmm_alloc_pages(n)`	Allocate n contiguous pages
`pmm_free_page(addr)`	Return page to free pool
`pmm_get_stats()`	Query usage statistics

Statistics tracked (append-only — never decrease): total pages, used pages, free pages, reserved pages, total lifetime allocations, total lifetime frees, peak usage high-water mark.

Virtual Memory Manager (VMM)

4-level x86_64 paging (PML4 → PDPT → PD → PT):

Function	Description
`vmm_map_page(virt, phys, flags)`	Map a virtual page to physical
`vmm_unmap_page(virt)`	Remove a page mapping
`vmm_get_physical(virt)`	Translate virtual to physical
`vmm_is_mapped(virt)`	Check if address is mapped

Page flags: `PTE_PRESENT`, `PTE_WRITABLE`, `PTE_USER`, `PTE_WRITETHROUGH`, `PTE_NOCACHE` (for MMIO), `PTE_HUGE` (2MB pages), `PTE_GLOBAL`, `PTE_NX`.

The first 1GB is identity-mapped (virtual == physical) via 2MB huge pages at boot. Framebuffer MMIO at high physical addresses (e.g., `0xFD000000`) is mapped explicitly with `PTE_NOCACHE | PTE_WRITETHROUGH`.

Heap Allocator

Kernel heap provides `kmalloc(size)` and `kfree(ptr)` for dynamic allocation within the identity-mapped region.

---

4. Interrupt Architecture

Component	Details
IDT	256 entries, all pointing to assembly stubs in `interrupts.S`
PIC	Dual 8259, remapped to vectors 32–47
Timer	PIT channel 0, ~100 Hz, drives scheduling and animation
Keyboard	IRQ1, scancode set 1, shift/ctrl/alt tracking
Mouse	IRQ12, PS/2 3-byte packets, cursor bounds clamping
ACPI SCI	IRQ9, power button events
Cascade	IRQ2 auto-unmasked when any slave IRQ (8–15) enabled

The PIT timer at ~100 Hz drives the entire event loop: input polling, animation ticks, key evolution, anomaly detection, and screen updates.

---

5. PCI Bus & Device Discovery

Full PCI bus enumeration via I/O ports `0xCF8` / `0xCFC`. Scans all buses, devices, and functions to discover:

- VirtIO devices (networking, GPU, console)
- USB UHCI controllers
- GPU devices (Intel, VMware SVGA, Bochs)

- PIIX4 power management
- Storage controllers (ATA)

The `lspci` shell command displays all discovered devices with vendor/device IDs and BARs.

---

6. Graphics Pipeline

Framebuffer

32bpp ARGB framebuffer obtained from Multiboot2 framebuffer info. Double-buffered with a backbuffer in kernel memory.

Constant	Value
Default resolution	1024×768
Color depth	32 bits per pixel (ARGB)
Supported resolutions	800×600, 1024×768, 1280×720, 1280×1024
Tile size (dirty tracking)	32×32 pixels
Max resolution	1280×1024

Drawing Primitives

Function	Description
`fb_put_pixel(x, y, color)`	Single pixel
`fb_fill_rect(x, y, w, h, color)`	Solid rectangle
`fb_draw_rect(x, y, w, h, color)`	Rectangle outline
`gfx_draw_line(x0, y0, x1, y1, color)`	Bresenham's line
`gfx_draw_hline(x, y, w, color)`	Fast horizontal line
`gfx_draw_vline(x, y, h, color)`	Fast vertical line
`gfx_alpha_blend(fg, bg, alpha)`	Per-channel alpha compositing
`gfx_fill_gradient_v(x, y, w, h, top, bot)`	Vertical gradient
`gfx_fill_rounded_rect(x, y, w, h, r, color)`	Rounded solid rect
`gfx_draw_rounded_rect(x, y, w, h, r, color)`	Rounded outline
`gfx_draw_shadow(x, y, w, h, offset, alpha)`	Drop shadow
`gfx_draw_text_scaled(x, y, str, color, scale)`	Scaled text

Alpha Blending

Per-channel formula: `result = (fg × alpha + bg × (255 − alpha)) / 255`

Used for window shadows, fade transitions, icon anti-aliasing, hover effects, and translucent overlays.

Dirty Rectangle Tracking

The framebuffer is divided into 32×32 pixel tiles. A bitfield tracks which tiles have been modified. In VM mode, only dirty tiles are uploaded to the display, reducing bandwidth. The tile grid supports up to 40 columns × 32 rows (1280×1024 at 32px tiles).

Font Rendering

Custom 8×16 bitmap font: - `font_draw_char(x, y, ch, fg, bg)` — single character - `font_draw_string(x, y, str, fg, bg)` — string rendering - Background color `0` = transparent (no fill behind character) - Bounds check: character rejected entirely if `y + 16 > framebuffer_height`

---

7. GPU Hardware Abstraction Layer

The GPU HAL provides a unified 2D acceleration interface with multiple backends. At boot, all backends register themselves. The highest-priority available backend is selected automatically.

Backends

Backend	Priority	Driver	Description
Intel BLT	100	`intel_gpu.c`	Intel integrated GPU BLT engine
VirtIO GPU	80	`virtio_gpu.c`	QEMU VirtIO GPU virtqueues
VMware SVGA	60	`vmware_svga.c`	VMware SVGA II FIFO protocol
Bochs VGA	40	`bochs_vga.c`	Bochs/QEMU VBE extensions
Software	0	built-in	CPU-only fallback (always available)

Operations

Operation	Description
`gpu_hal_fill_rect()`	Hardware-accelerated rectangle fill
`gpu_hal_clear()`	Full screen clear
`gpu_hal_copy_region()`	Screen-to-screen blit
`gpu_hal_flip()`	Backbuffer → frontbuffer swap
`gpu_hal_sync()`	Drain pending GPU operations
`gpu_hal_wait()`	Wait for GPU idle
`gpu_hal_set_resolution()`	Change display mode

If the active backend doesn't support an operation, it falls back to software rendering. Statistics track: fills, clears, copies, flips, batched ops, software fallbacks, bytes transferred.

VMware SVGA II Notes

- QEMU uses `SVGA_IO_MUL=1`: value port at BAR0+1 (not +4 as documentation suggests)
- QEMU does not implement `SVGA_CMD_RECT_FILL` or `SVGA_CMD_RECT_COPY`
- Uses CPU backbuffer + `SVGA_CMD_UPDATE` for display refresh

---

8. Window Manager

Up to 32 simultaneous windows with full compositing.

Window Properties

Property	Value
Max windows	32
Title bar height	28 pixels
Border width	1 pixel
Close button	18×18 pixels (red circle with X)
Corner radius	8 pixels
Shadow offset	4 pixels
Shadow opacity	100/255

Window Structure

Each window has: - Position (x, y), dimensions (width, height) - Title string (64 chars max) - Pixel buffer for content area - Four callbacks: `on_paint`, `on_click`, `on_key`, `on_close` - Drag

state (grab offset for title bar dragging) - Fade animation state (alpha 0–255, fading_in/fading_out flags) - Flags: visible, focused, dragging, closeable

Rendering Pipeline

1. Draw windows back-to-front (z-order maintained by list position)
2. For each window: a. Draw drop shadow (alpha-blended dark overlay, 4px offset) b. Draw rounded rectangle background (8px corner radius) c. Draw gradient title bar (dark → slightly lighter) d. Draw title text and close button e. Call `on_paint` callback to render content
3. Draw mouse cursor on top

Window Lifecycle

```
Create → Fade In → Active (paint/click/key) → Close → Fade Out → Destroy
```

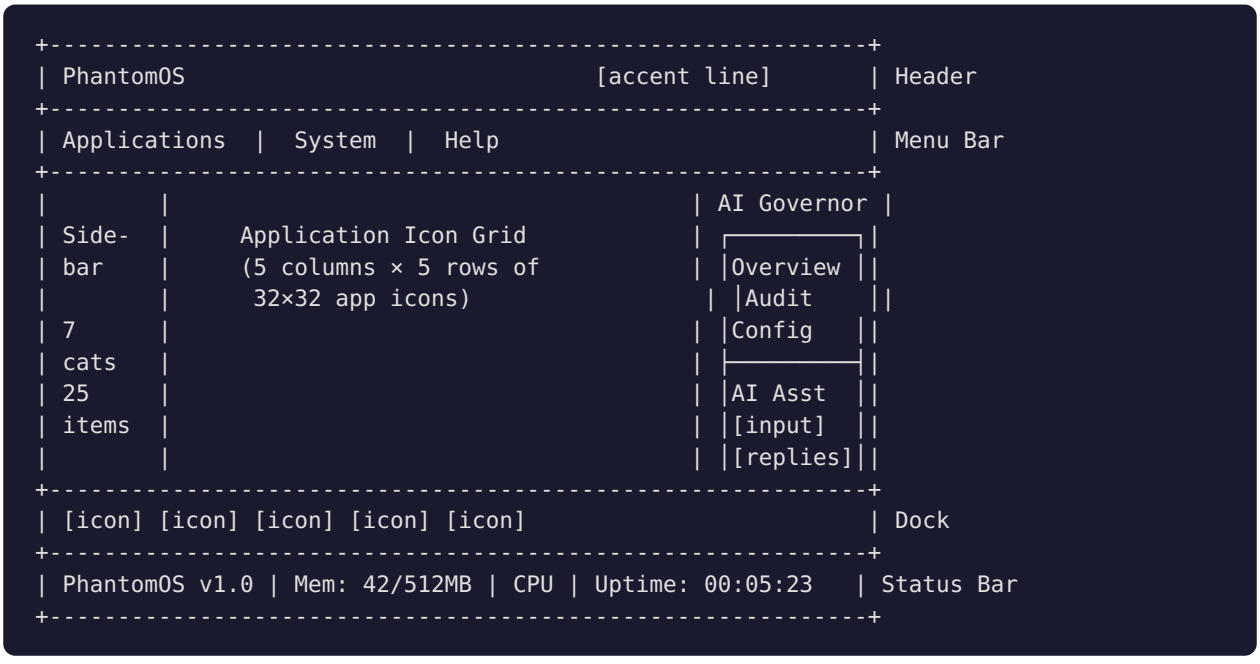
Each app uses a static guard variable (`static int xxx_win = 0`) to prevent duplicate windows. The `on_close` callback resets this guard to 0, allowing the window to be reopened later.

---

9. Desktop Environment

The desktop is the primary user interface, rendered in `desktop.c` (9,502 lines).

Layout



Sidebar (7 Categories, 25 Items)

#	Category	Items
1	System	System Monitor, Processes, Settings
2	Files	File Browser
3	Security	DNAuth, LifeAuth, BioSense, Governor, Constitution, PVE Encrypt
4	Network	QRNet, Network
5	Media	ArtOS, MusiKey, Media Player, PhantomPods
6	Tools	Terminal, Notes, Backup, Desktop Lab, GPU Monitor
7	Apps	Geology Viewer, Users

Each category has a mini-icon (8×8) and expands on click to reveal sub-items with smooth animation.

Visual Design

Element	Treatment
Windows	Rounded corners (8px), gradient title bars, drop shadows
Progress bars	Pill-shaped with rounded ends
Sidebar	Expand/collapse animation, mini-icons when collapsed
App icons	32×32, 16-color palettes, alpha-blended AA edges
Dock icons	16×16 with same AA treatment
Header	Accent line, text shadow
Panels	Card shadows
Status bar	Gradient fill
Hover effects	Color shift on interactive elements
Window transitions	Fade in on create, fade out on destroy

Theme Colors

Name	Hex	Usage
BG Dark	`#1A1A2E`	Desktop background
Accent	`#0F3460`	Title bars, panels
Highlight	`#E94560`	Active/critical elements
Text	`#EEEEEE`	Primary text
Text Dim	`#888888`	Secondary text
Header BG	`#0D1117`	Header/dock
Panel BG	`#111827`	Content panels
Green	`#22C55E`	Good/active status
Yellow	`#EAB308`	Warning/files
Purple	`#8B5CF6`	AI/creative
Orange	`#F97316`	Security

Event Loop

The desktop runs a continuous event loop at ~100 Hz (driven by PIT timer):

Per tick: 1. Poll mouse and keyboard input 2. Dispatch events to focused window (click/key callbacks) 3. Evolve PVE encryption key (if initialized) 4. Run Governor anomaly detection 5. Update threat timeline (24-slot circular buffer) 6. Expire stale alerts (timeout: 3000 ticks = 30 seconds) 7. Advance animation ticks for all active apps 8. Repaint all dirty windows 9. Flip backbuffer to screen

---

10. Widget System

8 reusable UI widget types used across all applications:

Widget	Description	Key Constants
Button	Clickable rectangle with label	Configurable size, color
List	Scrollable item list with selection	Max 64 items, 20px item height
Text Input	Editable text field with cursor	128 char max, focus tracking
Progress Bar	Filled/empty bar	0–100 value range
Scrollbar	Vertical scroll with up/down arrows	14px width, 14px arrow buttons
Checkbox	Toggle with label	Checked/unchecked state
Tab Bar	Horizontal tab selector	Max 8 tabs, 24px height
Label	Static text display	Foreground/background colors

11. Icon Rendering

System

Each app has three icon sizes: - **App icons**: 32×32 pixels (main grid and windows) - **Dock icons**: 16×16 pixels (bottom dock bar) - **Sidebar icons**: 8×8 pixels (collapsed category indicators)

Palette System

Icons use 16-color indexed palettes optimized per icon (not a shared global palette). Each icon's palette is hand-tuned for its specific colors and shapes.

Anti-Aliasing

Edge pixels store alpha values (e.g., 128, 192) for smooth transitions. These are composited over the background via `gfx_alpha_blend()`, producing crisp icons with smooth boundaries — no jagged edges despite the small pixel dimensions.

12. Applications (25 Total)

Fully Interactive (paint + click + keyboard)

File Browser (460×400)

- Directory navigation with path bar
- Filename filter (type to search)
- File preview pane (2KB preview of file content)
- Scrollbar for long listings (64 files max, 128-char names)
- Navigation history (16 entries)
- Tab system: Files tab and Views tab (32 view slots)
- Dialog modes: mkdir, new file, rename, copy, hide, snapshot

Terminal (500×360)

- Shell I/O with 16KB scrollback buffer
- Command input (256 chars max)
- Command history (16 entries, up/down arrow navigation)
- Cursor display and scrollbar
- Executes all 31 shell commands

ArtOS (420×480)

- Canvas: 128×96 pixels with configurable pixel scale
- 8 drawing tools: Pencil, Line, Rect, FillRect, Ellipse, Fill, Eraser, Eyedrop

- 16-color palette with click selection
- 32-level undo stack (full canvas snapshots)
- **AI Art Generator**: 64-char text prompt → procedural art generation
- **DrawNet**: Collaborative drawing UI (local-only mode)

MusiKey (520×400)

- Piano keyboard interface with clickable keys
- Authentication enrollment mode (record key sequences)
- Sound visualizer bars
- Key sequence verification

Notes (400×360)

- 8 notes max, 256 chars each, 32-char titles
- Note list with selection
- Text editing area with cursor
- New/Save buttons
- Append-only (notes are never deleted)

PVE Encrypt (300×330)

- Planck Variable Encryption with PVE-SBC cipher mode
- Live evolving key display (16 bytes in hex, updates every tick)
- Bar chart history of key evolution (32 slots)
- Text input for plaintext (64 chars max)
- Encrypt/Decrypt buttons
- Ciphertext hex display (block-aligned, up to 80 bytes)
- Decrypted text display
- Mode label: "Mode: PVE-SBC [N blocks]"
- Governor-audited operations
- See Section 14 for cipher details

Interactive (paint + click)

DNAuth (300×440)

- DNA sequence display (32 characters of A/C/G/T bases)
- Enrollment and verification workflows
- Scan animation (progress += 2 every 3 ticks)
- Match confidence percentage
- Status messages with color coding

LifeAuth (300×420)

- Vital signs monitoring: heart rate (72 bpm default), SpO2 (98%), plasma level (94%), body temperature (36.9°C)
- Progress bars for each vital
- Scan animation (progress += 3 every 3 ticks)

- Vitals randomize on scan completion

BioSense (320×460)

- Vein density bar chart (8 zones, 0–100 each)
- Scan and authenticate buttons
- Scan animation (progress += 2 every 3 ticks)
- Match percentage generation

QRNet (300×460)

- 16×16 QR code grid visualization
- Peer ID display (16 characters)
- Packet counter (increments every 50 ticks when connected)
- Connect/Disconnect toggling
- Generation animation (progress += 5 every 2 ticks)

Media Player (300×440)

- Playlist with 6 tracks
- Transport controls: play, pause, stop, previous, next
- Track progress bar (advances every 5 ticks)
- 24-bar audio visualizer (randomizes every 3 ticks)
- Auto-advance to next track on completion
- Volume control (0–100)

Users (340×340)

- User list with selection
- Detail panel showing role, state, authentication method
- Click to select different users

Backup (320×400)

- Snapshot history list
- Backup progress bar (advances every 3 ticks)
- Time travel to previous snapshots
- New snapshot entry created on backup completion

Desktop Lab (300×380)

- Theme cycling (multiple color schemes)
- Accent color cycling
- Scale cycling
- Live preview of all changes

Tabbed (paint + click + keyboard)

Geology Viewer (580×440)

- **Strata Tab**: Color-coded geological strata bands representing filesystem layers over time
- **Branches Tab**: Branch list with diff information between branches
- **Dashboard Tab**: Storage gauges showing content/ref/view region usage percentages
- Keyboard navigation between tabs

Display-Only (paint)

App	Size	Description
System Monitor	260×300	CPU, memory, uptime, PIT tick count
Settings	280×280	System configuration display
Security	300×360	Security policy status overview
Processes	280×280	Running process list
Governor	300×380	AI Governor status (see Section 13)
Constitution	320×400	System constitution / rules
Network	280×280	Interface status, IP, MAC, TX/RX, ping results
PhantomPods	360×380	Pod/container status display
GPU Monitor	280×420	Active GPU backend, operation stats, fallback counts

---

13. AI Governor (v1–v4)

The AI Governor is the policy enforcement engine that implements the "To Create, Not To Destroy" philosophy. It sits between application requests and kernel operations, intercepting every potentially destructive action.

Policy Domains

Domain	Flag	Scope
Memory	`0x0001`	Memory freeing, overwriting
Process	`0x0002`	Process kill, termination
Filesystem	`0x0004`	Delete, truncate, overwrite
Resource	`0x0008`	Resource exhaustion

Policy Types (13 Total)

Policy	Default Verdict	Behavior
`MEM_FREE`	ALLOW (kernel) / DENY	Memory deallocation
`MEM_OVERWRITE`	AUDIT	Logged but allowed
`PROC_KILL`	DENY (strict) / AUDIT	Forcible termination blocked
`PROC_EXIT`	ALLOW	Graceful self-termination permitted
`FS_DELETE`	**TRANSFORM**	Delete → hide (Prime Directive)
`FS_TRUNCATE`	DENY	Data destruction blocked
`FS_OVERWRITE`	AUDIT	GeoFS preserves history anyway
`FS_HIDE`	ALLOW	This IS the safe alternative
`FS_PERM_DENIED`	DENY	Permission failure
`FS_QUOTA_EXCEEDED`	DENY	Over quota
`RES_EXHAUST`	DENY	Resource exhaustion attempt

Verdicts

Verdict	Meaning
`GOV_ALLOW`	Operation permitted
`GOV_DENY`	Operation blocked
`GOV_TRANSFORM`	Converted to safe alternative (e.g., delete → hide)
`GOV_AUDIT`	Allowed but logged as suspicious

Threat Levels

Level	Value	Trigger
NONE	0	Normal operation
LOW	1	Default assessment
MEDIUM	2	Network/exec patterns detected
HIGH	3	Delete/destroy patterns detected
CRITICAL	4	Format/truncate/overwrite patterns detected

v1: Basic Policy Engine

- Policy check and enforcement for all 4 domains
- Circular audit buffer (1024 entries)
- Capability-based access control (`GOV_CAP_*` flags)
- Per-policy enable/disable

v2: Predictive Intelligence

- Predictive threat trends (rising / falling / stable) over 12-slot sliding window
- Health score (0–100): starts at 100, −10 per warning, −20 per critical
- Decision explanations appended to audit entries for TRANSFORM and DENY verdicts
- Interactive 8-page system tutorial covering all subsystems

v3: Anomaly Detection

Anomaly	Threshold	Description
Memory spike	>20% increase	Sudden allocation surge
Process surge	>5 new processes	Rapid process creation
Violation burst	>3 violations in 5 sec	Repeated policy violations
Rapid denials	>2 new denials	Cluster of blocked operations
Memory bomb	≥3 MEM_FREE denials in 10 sec	Malicious pattern
Fork bomb	≥3 PROC_KILL denials	Malicious pattern
Mass deletion	≥3 FS_DELETE transforms	Malicious pattern
Resource exhaustion	≥1 RES_EXHAUST denial	Malicious pattern

Context-aware AI assistant responses tagged with: `[!ALERT]`, `[MEM HIGH]`, `[HEALTH LOW]`

Alert expiration: 3000 ticks (30 seconds)

v4: Behavioral Learning

- **Per-policy normalcy profiles**: Tracks baseline allow/deny/transform counts for each policy
- **Deviation detection**: Flags operations that deviate significantly from learned norms
- **Threat timeline**: 24-slot bar chart showing threat + health history over time
- **Smart recommendations**: Actionable suggestions based on current system state
- **Quarantine system**: 8 slots for capturing suspicious operations
- Capture → review → release workflow
- Each entry stores: policy, verdict, PID, timestamp, reason (48 chars), reviewed flag

Governor GUI (3 Tabs)

- **Overview**: Threat level, health score bar, active anomalies, threat timeline chart
- **Audit**: Scrollable log of policy decisions with explanations
- **Config**: Interactive checkboxes to enable/disable policies, per-policy statistics

AI Assistant Panel

Located in the right panel of the desktop: - Text input for natural-language questions - Fuzzy string matching for query interpretation - Context-aware responses drawing from real system

state - Recognizes: memory pressure, health drops, active alerts, threat level changes - Maximum 6 active alerts displayed simultaneously

---

14. PVE-SBC Encryption

Planck Variable Encryption — S-box, Block Chaining combines three cryptographic techniques into a single cipher mode.

Constants

Constant	Value
Key length	16 bytes
Block size	16 bytes
Max plaintext	64 bytes
Max ciphertext	80 bytes (includes PKCS#7 padding)
History slots	32 (for key evolution visualization)

Key Evolution

The encryption key evolves continuously via an LCG seeded with: - PIT timer ticks - Planck clock counter (internal monotonic counter) - Address-space entropy

When the user clicks "Encrypt," the current key is snapshot for that operation.

Encryption Pipeline (per 16-byte block)

```
Plaintext
  → PKCS#7 Padding (pad to 16-byte boundary; pad_amt = 16 − (len % 16))
  → CBC Chain (XOR block with previous ciphertext block, or IV for first)
  → S-box Substitution (AES Rijndael 256-byte lookup table)
  → Key Stream XOR (LCG-generated unique byte per position)
  → Ciphertext
```

Decryption Pipeline (per 16-byte block)

```
Ciphertext
  → Key Stream XOR (same LCG stream)
  → Inverse S-box (reverse Rijndael table)
  → CBC Unchain (XOR with previous ciphertext, or IV)
  → Strip PKCS#7 Padding
  → Plaintext
```

Component Details

S-box: Standard AES Rijndael S-box — 256-byte nonlinear substitution. No fixed points. Breaks the linear relationship that makes XOR trivially attackable.

CBC (Cipher Block Chaining): Each plaintext block XORed with previous ciphertext before encryption. First block uses IV. Identical plaintext blocks produce different ciphertext.

Key Stream Generator: LCG expands 16-byte key into unique byte stream: - State seeded by XOR-folding key bytes into 64-bit value - `state = state × 6364136223846793005 + 1442695040888963407` - Output: `stream[i] = (uint8_t)(state >> 33)`

IV Derivation: Separate LCG with different additive constant (`7046029254386353131`) generates 16-byte IV deterministically from the snapshot key.

PKCS#7 Padding: Pad to 16-byte boundary. Each padding byte contains the pad amount. Full blocks get 16 bytes of padding.

Governor Integration

All encrypt/decrypt operations are audited: - Encrypt: policy `POLICY_RES_EXHAUST`, verdict `GOV_ALLOW`, reason `"PVE-SBC encrypt"` - Decrypt: policy `POLICY_RES_EXHAUST`, verdict `GOV_AUDIT`, reason `"PVE-SBC decrypt"`

---

15. GeoFS Filesystem

GeoFS (Geological Filesystem) is an append-only, immutable-layered, content-addressed filesystem. Inspired by geological strata — data accumulates in layers, nothing is ever removed.

Design Principles

- **Write** = deposit a new stratum
- **Delete** = hide beneath a new view layer (data remains intact)
- **Branch** = tectonic divergence (parallel timelines)
- **View** = geological survey (see a specific point in time)

Storage Regions

Region	Default Size	Purpose
Content	256 pages (1MB)	File data with SHA-256 deduplication
References	64 pages (256KB)	Path → content hash mappings
Views	32 pages (128KB)	View/branch metadata

Constants

Constant	Value
Block size	4096 bytes
Hash algorithm	SHA-256 (32 bytes)
Max path	512 bytes
Max filename	128 bytes
Branch name max	64 bytes
Hash buckets	256 (for dedup lookup)
Max ancestry depth	256
Dir marker	`"__PHANTOM_DIR__"`

Content-Addressable Deduplication

File data is stored by SHA-256 hash. Writing identical data twice reuses the existing content block. A 256-bucket hash table with chaining provides fast content lookup.

```
Write "hello" to /a.txt  →  Content stored, hash H1, ref /a.txt → H1
Write "hello" to /b.txt  →  Hash H1 already exists, ref /b.txt → H1 (no data duplication)
```

View System

Views are named snapshots of filesystem state: - `kgeofs_view_create(vol, "v1.0", &id)` — create a named checkpoint - `kgeofs_view_switch(vol, id)` — see the filesystem at that point - `kgeofs_view_hide(vol, path)` — hide a file in current view (data preserved)

Views form a linear history chain. The current view determines which files are visible.

Branch System

Branches represent parallel filesystem timelines (tectonic divergence): - Each branch tracks its parent and creation view - Merge operation combines two branch histories - Conflict detection via ancestry comparison - Maximum ancestry depth: 256 levels

Volume Persistence

Volumes can be saved to and loaded from ATA disk: - Persist magic: `0x504852534F45474B` ("KGEOFPHR") - Persist version: 2 (with branch support) - Shell commands: `save` (to disk), `load` (from disk)

Governor Integration

The Governor enforces append-only guarantees at the policy level: - `FS_DELETE` → TRANSFORM to `view_hide()` (data preserved) - `FS_TRUNCATE` → DENY (would destroy data) -

`FS_OVERWRITE` → AUDIT (GeoFS appends new version anyway) - `FS_HIDE` → ALLOW (this is the correct safe operation)

Error Codes

Code	Name	Meaning
0	OK	Success
−1	ERR_IO	I/O error
−2	ERR_NOMEM	Out of memory
−3	ERR_NOTFOUND	Path not found
−4	ERR_EXISTS	Already exists
−5	ERR_INVALID	Invalid argument
−6	ERR_FULL	Region full
−7	ERR_CORRUPT	Data corruption
−8	ERR_ISDIR	Path is directory
−9	ERR_NOTDIR	Not a directory
−10	ERR_PERM	Permission denied
−11	ERR_QUOTA	Quota exceeded
−12	ERR_CONFLICT	Branch conflict

---

16. VirtIO Networking

VirtIO-net PCI device driver providing Layer 2/3 networking.

Configuration

Parameter	Value
IP address	`10.0.2.15/24` (static)
Gateway	`10.0.2.2` (QEMU user-mode)
Queue size	64 entries per virtqueue
RX buffer	1526 bytes (10 virtio hdr + 14 eth + 1500 MTU + 2 pad)
ARP table	16 entries with timeout

Protocol Stack

Layer	Protocol	Implementation
L2	Ethernet	Frame construction/parsing, MAC address
L2.5	ARP	Request/reply, 16-entry cache, timeout
L3	IP	Header construction, checksum
L3	ICMP	Echo request (ping), echo reply (respond to pings)

ARP

- Responds to ARP requests for our IP with our MAC
- Sends ARP requests to resolve gateway and target IPs
- 16-entry cache with expiration
- Required for any IP communication

ICMP

- **Echo Reply**: Automatically responds to incoming pings (type 0)
- **Echo Request**: `virtio_net_ping(dest_ip, seq)` sends ping (type 8)
- Checksum calculation and validation
- `ping` shell command sends 4 packets with round-trip timing

VirtIO Ring Buffers

Two virtqueues (RX and TX), each with 64 descriptor entries: - **RX ring**: Device fills with incoming packets, driver processes - **TX ring**: Driver submits outgoing packets, device transmits - Doorbell notification via I/O port write to signal device

Network Window

Live desktop window displaying: MAC address, IP address, link status, TX/RX packet counts, bytes transferred, ping results.

---

17. USB HID Stack

UHCI Host Controller (`usb.c`)

USB 1.1 Universal Host Controller Interface:

Parameter	Value
USB version	1.1
Controller	UHCI (PCI class 0x0C, subclass 0x03, prog-if 0x00)
Frame list	1024 entries
Ports	2
Max packet	64 bytes

Discovery via PCI enumeration. I/O port base from BAR4. Supports control transfers (device enumeration) and interrupt transfers (periodic HID polling).

HID Boot Protocol (`usb_hid.c`)

Parameter	Value
Max HID devices	4
Device types	Keyboard (1), Mouse (2)
Report size	8 bytes (boot protocol)
Polling	Via frame list scheduled TDs

Device enumeration: 1. Port reset and enable 2. SET_ADDRESS (assign unique address) 3. GET_DESCRIPTOR (device, config, interface, HID, endpoint) 4. SET_CONFIGURATION (activate device) 5. SET_PROTOCOL (boot mode — simplified 8-byte reports) 6. Schedule interrupt IN transfers for periodic polling

Keyboard: Scancode translation from USB HID usage IDs. Press/release detection by comparing current report with previous report. Modifier key tracking.

Mouse: Relative X/Y movement from report bytes 1–2, button state from byte 0. Integrated with PS/2 mouse cursor system.

---

18. ACPI Power Management

PIIX4 PM device support for QEMU i440fx platform:

Feature	Details
Device	PIIX4 PM (PCI vendor 0x8086, device 0x7113)
SCI IRQ	IRQ9 (power button events)
PM I/O base	Read from PCI config offset 0x40
Sleep control	PM1a_CNT register

Operations

Function	Description
`acpi_init()`	Detect PIIX4, configure SCI on IRQ9
`acpi_request_shutdown()`	Set shutdown flag (checked by event loop)
`acpi_poweroff()`	Write S5 sleep type to PM1a_CNT → power off
`acpi_reboot()`	Write 0x06 to port 0xCF9 → system reset

The shutdown flow: power button press → IRQ9 → SCI handler → set shutdown flag → event loop checks → graceful cleanup → `acpi_poweroff()`.

---

19. VM Detection & Optimization

Hypervisor Detection

CPUID leaf `0x40000000` identifies the hypervisor: - **KVM**: Signature "KVMKVMKVM" → enable paravirt clock - **QEMU**: TCG mode → enable dirty-rect optimization - **VMware**: Signature "VMwareVMware" → enable SVGA II driver - **Hyper-V**: Signature → detected but no special handling - **Xen**: Signature → detected but no special handling

KVM Paravirtualized Clock

MSR-based high-resolution clock: - MSR `0x4B564D01` (KVM_SYSTEM_TIME_NEW) - Shared memory page with TSC multiplier and offset - Provides nanosecond-precision timestamps without VM exits

VM Optimizations

- **Dirty-rect tile tracking**: Only upload modified 32×32 pixel tiles
- **Timer-based frame limiting**: Avoid busy-loop rendering
- **VSync**: Synchronize buffer flips with display refresh

---

20. Shell Interface

31 commands available in the text-mode shell (also accessible via Terminal window):

Filesystem Commands

Command	Description
`ls [path]`	List directory contents
`cat <file>`	Display file contents
`write <file> <data>`	Write data to file
`append <file> <data>`	Append data to file
`mkdir <path>`	Create directory
`hide <path>`	Hide file (not delete — Governor enforced)
`pwd`	Print working directory
`cd <path>`	Change directory
`stat <file>`	Show file metadata
`mv <src> <dst>`	Move/rename file
`cp <src> <dst>`	Copy file
`tree [path]`	Show directory tree
`find <pattern>`	Search for files
`chmod <perms> <file>`	Change permissions
`chown <owner> <file>`	Change ownership

GeoFS Commands

Command	Description
`views`	List all views
`view <id>`	Switch to view
`snapshot <name>`	Create named view
`diff <view1> <view2>`	Compare views
`branch <name>`	Create branch
`merge <branch>`	Merge branch

Persistence Commands

Command	Description
`export`	Write volume to ATA disk
`import`	Load volume from ATA disk
`save`	Save volume state
`load`	Restore volume state

System Commands

Command	Description
`help`	List all commands
`clear`	Clear screen
`mem`	Memory statistics
`disk`	Disk usage
`gov`	Governor statistics
`uptime`	System uptime
`echo <text>`	Print text
`exit`	Exit shell

Hardware Commands

Command	Description
`lspci`	List PCI devices
`gpu`	GPU HAL info
`usb`	USB device listing
`net`	Network status
`ping [ip]`	Send ICMP echo (4 packets)

Access Control

Command	Description
`su <user>`	Switch user
`whoami`	Current user
`quota`	Disk quota info
`ps`	Process listing

---

21. Codebase Statistics

Source File Sizes

File	Lines	Description
`kernel/desktop.c`	9,502	Desktop GUI + all 25 apps
`kernel/geofs.c`	2,751	GeoFS filesystem
`kernel/shell.c`	1,575	Shell commands
`kernel/virtio_net.c`	1,008	VirtIO networking
`kernel/wm.c`	601	Window manager
`kernel/governor.c`	598	Policy engine
`kernel/icons.c`	598	Icon bitmaps + palettes
`kernel/widgets.c`	592	UI widget system
`kernel/framebuffer.c`	554	Framebuffer + dirty tracking
`kernel/governor_sim.c`	418	Behavioral learning
`kernel/graphics.c`	350	Drawing primitives
Other kernel files	~17,000	Boot, drivers, memory, etc.
Total	**~36,500**	**Complete bare-metal OS**

Compilation

```
Compiler:  GCC (x86_64, cross or system)
Flags:     -ffreestanding -fno-builtin -nostdlib -nostdinc
           -mno-red-zone -mcmodel=kernel -fno-pic
           -mno-sse -mno-sse2 -mno-mmx -mno-80387
           -Wall -Wextra -O2 -g
Assembler: GNU as (--64)
Linker:    GNU ld with custom linker script
Output:    phantomos.elf → phantomos.iso (via grub-mkrescue)
```

Constraints

Constraint	Reason
No SSE/FPU	FPU not initialized in kernel
No red zone	Interrupt handlers would corrupt it
No libc	Freestanding — custom strlen, memcpy, kmalloc
No snprintf	Use strcpy/strncpy + manual string building
No floating point	All math is integer-only
First 1GB identity-mapped	Heap and GeoFS use virt==phys
Kernel must fit in <16MB	Practical limit from linker layout

PhantomOS — "Nothing is ever truly lost."