# Chronos OS: Phase 15 & 16 Detailed Report

Bidirectional IP Networking and
Secure User Mode Application Execution

Development Log

January 20, 2026

**Abstract**

Following the successful implementation of the foundational network stack in earlier phases, Chronos OS entered a critical stage of evolution. This report documents the transition from a passive, kernel-only execution environment to a bidirectional, network-addressable operating system capable of securely executing user applications. The work presented addresses two major challenges: enabling reliable inbound network communication and establishing a protected execution boundary between kernel and user space. Key outcomes include the resolution of ARP-induced routing deadlock, the introduction of a hierarchical page table permission override mechanism, and the deployment of a functional ELF binary loader. Collectively, these developments mark Chronos OS's transformation from a kernel experiment into a viable application platform.

## Contents

# 1 Introduction

With basic device drivers and outbound packet transmission already functional, Chronos OS reached a developmental inflection point. To progress toward the goal of a daily-usable operating system, two capabilities became mandatory: external communication with the network and a secure execution model for non-kernel programs. This report consolidates the engineering decisions and architectural corrections required to achieve these goals during Phases 15 and 16.

# 2 Phase 15: Bidirectional IP Networking

## 2.1 The ARP Deadlock Phenomenon

Initial attempts to establish IP connectivity appeared partially successful. Chronos OS was able to transmit ICMP Echo Requests to the QEMU gateway at `10.0.2.2`. However, no corresponding Echo Replies were observed, despite correct packet construction.

**Root Cause:** Modern IP routing depends on a populated Address Resolution Protocol (ARP) cache. Before the gateway could respond to Chronos, it needed to resolve the physical MAC address associated with Chronos's assigned IP (`10.0.2.15`). Because the initial network stack ignored inbound ARP Requests, the gateway never received this information. As a result, all inbound IP traffic destined for Chronos was silently dropped.

## 2.2 Reactive ARP Response Implementation

To resolve this deadlock, an ARP responder was implemented. When the network driver detects an Ethernet frame with EtherType `0x0806`, the packet is dispatched to the ARP handler. The handler validates the target IP address and, if it matches the OS identity, constructs and transmits a compliant 60-byte ARP Reply.

This reactive behavior allows the gateway to successfully populate its ARP cache, thereby enabling inbound IP traffic for the first time.

## 2.3 ICMP Connectivity Verification

With ARP resolution functioning, ICMP connectivity was re-tested. A `ping` utility was implemented that constructs a complete 74-byte Ethernet frame containing IP and ICMP headers. Both headers utilize the standard 16-bit One's Complement checksum algorithm to ensure packet validity.

**Result:** The gateway successfully responded with ICMP Echo Replies, confirming the establishment of reliable bidirectional IP communication.

# 3 Phase 16: ELF Loading and Hierarchical Memory Protection

## 3.1 Phase Objective

The objective of Phase 16 was to execute externally compiled ELF (Executable and Linkable Format) binaries in Ring 3 (User Mode). Achieving this required a controlled transition from Ring 0 execution and the enforcement of strict memory access boundaries.

### 3.2 The Hierarchical Gating Barrier

Early attempts to execute user applications resulted in immediate `PROTECTION_VIOLATION` page faults.

**Architectural Cause:** On x86_64 systems, memory access permissions are enforced hierarchically across a four-level paging structure. Even when a final page table entry is marked as user-accessible, the CPU will deny access if any parent entry (PML4, PDPT, or PD) remains supervisor-only.

**Resolution:** A manual page table traversal routine was implemented. During user process creation, the kernel now walks the page table hierarchy and explicitly sets the USER bit on all intermediate entries leading to the application's virtual address range. This guarantees an uninterrupted permission path from the PML4 root to the leaf page.

### 3.3 Reconciling Physical Address Spaces

A secondary fault source was traced to incorrect physical address translation for the user stack. Chronos OS maintains two distinct virtual-to-physical translation schemes:

1. **HHDM Region:** Used for dynamically loaded modules, translated via `Virt - HHDM_OFFSET`.
2. **Kernel Binary Region:** Used for static kernel data, translated via `Virt - KERNEL_DELTA`.

Applying the incorrect translation method to the user stack resulted in mappings to invalid physical memory. Correcting this distinction eliminated early stack-related page faults.

### 3.4 System Call Boundary Verification

To verify controlled privilege transitions, a basic system call interface was introduced using `int 0x80`. User applications are prohibited from directly accessing hardware resources and must request services via this interrupt.

Successful handling of the interrupt by the kernel confirmed a clean transition from Ring 3 to Ring 0 and validated the enforcement of privilege separation.

## 4 Verification and Results

End-to-end verification was performed using an externally compiled Rust application packaged as `testapp.elf` and loaded from the ramdisk.

1. The shell invokes `run testapp`.
2. The kernel loads and maps the ELF binary at `0x400000`.
3. Page table permissions are updated for user access.
4. Execution transitions to Ring 3 via `iretq`.
5. The application triggers a system call using `int 0x80`.
6. The kernel logs successful receipt of a Ring 3 system call.

## 5 Conclusion

Phases 15 and 16 represent a structural shift in the Chronos OS architecture. The operating system now possesses a verified network identity and a protected execution environment for user applications. These capabilities establish the foundation for higher-level services, graphical interfaces, and multitasking in subsequent development phases.