Chronos OS: Phase 9 Report

Virtual Memory Management and Page Table Manipulation

Development Log

January 20, 2026

Abstract

Following the successful transition to Ring 3 (User Mode) in Phase 8, Chronos OS encountered the x86_64 Memory Protection unit. User Mode code is forbidden from accessing pages marked as "Supervisor Only." Phase 9 focused on the implementation of a Virtual Memory Manager (VMM) capable of interacting with the 4-Level Page Tables. We utilized the Higher Half Direct Map (HHDM) protocol to locate physical page tables and successfully modified the Page Table Entries (PTEs) to allow user execution of specific memory regions.

Contents

1 The Memory Protection Barrier

Upon entering Ring 3, the CPU enforces Paging permissions strictly. The ‘user$_{m}ode_{a}pp‘$ $functionresidesintheke$

$Executingcodeinthisregionfrom Ring 3causesa$**Page Fault (Exception 14)**$withthe$`PROTECTION_VIOLATION_`

2 VMM Implementation

2.1 HHDM (Higher Half Direct Map)

To modify a Page Table, the CPU needs its Virtual Address. However, the CR3 register only holds the Physical Address of the PML4 (Page Map Level 4).

We utilized the Limine `HhdmRequest`. This maps all of physical RAM to a known virtual offset (e.g., `0xFFFF800000000000`).

$$VirtAddr = PhysAddr + HHDM\_Offset$$

This allows the kernel to treat Page Tables as standard Rust structs in memory.

2.2 Bit Manipulation

We implemented a function ‘mark$_{a}s_{u}ser‘$$thattraversesthepagetablesforaspecificaddress.$

```rust
pub fn mark_as_user(mapper: &mut OffsetPageTable, address: u64) {
    let page = Page::containing_address(VirtAddr::new(address));

    // Set the USER_ACCESSIBLE bit (Bit 2)
    // This tells the CPU: "Ring 3 is allowed here."
    let flags = PageTableFlags::PRESENT
              | PageTableFlags::WRITABLE
              | PageTableFlags::USER_ACCESSIBLE;

    unsafe {
        mapper.update_flags(page, flags).unwrap().flush();
    }
}
```

Listing 1: Updating Page Flags

3 Integration and Verification

3.1 The Execution Flow

1. **Boot:** Kernel initializes in Ring 0. 2. **VMM Init:** Kernel requests HHDM offset and initializes the mapper. 3. **Unlock:** The kernel calculates the address of ‘user$_{m}ode_{a}pp‘$$andcalls‘mark_{a}s_{u}ser‘.$
$*Transition : **Thekernelexecutes‘iretq‘toswitchtoRing3.$

3.2 Results

Prior to Phase 9, the transition resulted in a Page Fault panic.
After implementing the VMM logic, the system logs:

```
[INFO] Unlocking Memory Page for User Mode...
[ OK ] Page Tables Updated
[INFO] Jumping to Ring 3...
```

The system then enters a stable state (freeze) inside the user loop. This confirms that the CPU is executing instructions in Ring 3, reading from memory that was dynamically unlocked by the kernel.

4 Conclusion

Chronos OS now possesses the two fundamental pillars of a modern secure operating system: **Privilege Separation (Ring 0 vs 3)** and **Virtual Memory Control**. The kernel can now sandbox applications, ensuring that a crash in a user program cannot bring down the entire system.