OBSIDIAN CIRCUIT

QWAMOS

QUBES + WHONIX ADVANCED MOBILE OPERATING SYSTEM

COMPREHENSIVE TECHNICAL SPECIFICATION

Version	3.1.0 (January 2026)	**Architecture**	ARM64-native KVM Hypervisor	**Primary Language**	Python 3.9+, Dart/Flutter, TypeScript
Status	27/27 Phases Complete — 100%	**License**	AGPL-3.0 (Open Source, Copyleft)	**Maintainer**	Desirae Stark
Codebase	50,000+ Lines of Code	**Repository**	github.com/Dezirae-Stark/QWAMOS	**Date**	April 2026

This document constitutes the authoritative technical specification for QWAMOS v3.1.0 (Qubes+Whonix Advanced Mobile Operating System). QWAMOS is the world's first mobile hypervisor operating system for ARM64 devices — providing intelligence-grade security, post-quantum cryptography, and Qubes-style VM compartmentalization on ARM64 smartphones.

SECTION 1 — WHAT IS QWAMOS?

World's First Mobile Hypervisor Operating System

QWAMOS (Qubes + Whonix Advanced Mobile Operating System) is the **world's first mobile hypervisor operating system** for ARM64 devices. It is a post-quantum secure, compartmentalized OS that boots as its own operating system on smartphones, turning them into intelligence-grade secure computing devices — functionally equivalent to running Qubes OS on a desktop, but on a phone.

Core Identity

Full Name	Qubes+Whonix Advanced Mobile Operating System
Version	3.1.0 (January 2026)
Architecture	ARM64-native hypervisor OS
License	AGPL-3.0 (open-source, copyleft)
Primary Maintainer	Desirae Stark
Development Phases	27 / 27 (100% complete)
Lines of Code	50,000+
Primary Language	Python 3.9+, Dart/Flutter, TypeScript, Java/Kotlin
Repository	github.com/Dezirae-Stark/QWAMOS

Mission & Design Principles

Principle	Implementation
Default-deny isolation	VMs cannot communicate unless explicitly permitted — no cross-VM data flow by default
Zero direct internet	All traffic exits only via the anonymizing Gateway VM — no user VM has direct internet access
Crypto-first storage	All data encrypted at rest with post-quantum algorithms and hardware-backed key storage
Defense-in-depth	12 overlapping security categories with no single point of failure across the full stack
Open-source transparency	AGPL-3.0; reproducible builds with signed manifests; SLSA Level 3 supply chain provenance
Adversary-grade hardening	Designed for Tier 4 (nation-state) threat actors; defeats advanced persistent threats
AI-powered defense	Triple-AI coordination with three independent ML models providing real-time threat detection

System Architecture Overview

SECTION 2 — HYPERVISOR & VM SYSTEM

KVM Backend, VM Domains & Resource Isolation

2.1 KVM Backend

QWAMOS uses Linux KVM with ARM64 VHE (Virtualization Host Extensions) as the hypervisor engine. QEMU provides device emulation with VirtIO drivers for optimal I/O performance. The hypervisor boots at EL2 (Exception Level 2), giving QWAMOS full hardware virtualization capabilities on supported ARM64 devices.

Component	File	Purpose
KVM Backend	hypervisor/kvm_backend.py	Hardware virtualization driver — EL2 management
KVM Manager	hypervisor/kvm_manager.py	VM lifecycle (create, start, stop, snapshot, migrate)
VirtIO Drivers	hypervisor/drivers/	Optimized I/O device emulation for all VM types
NIC Enforcer	hypervisor/nic_enforcer.py	Per-VM network interface policies — hardware-enforced routing
Resource Monitor	hypervisor/resource_monitor.py	CPU, memory, I/O telemetry per-VM in real time
GPU Manager	hypervisor/gpu_manager.py	Per-VM GPU slice allocation and Vulkan passthrough
GPU Policy	hypervisor/gpu_security_policy.py	GPU access control rules — prevents cross-VM GPU access
AI Governor	hypervisor/ai_governor.py	ML-driven resource scheduling and predictive VM scaling

2.2 VM Domain Architecture

VM Domain	Network	Base OS	Purpose
Dom0 / Control VM	None	Minimal Linux	VM management, security policies, ML monitoring — privileged, no network
Gateway VM	Direct (filtered)	Whonix-style Linux	Tor/I2P routing; all other VMs route through here — mandatory anonymization gateway
Android VM	Via Gateway	AOSP Android 14+	Daily use Android environment — primary user interface
Kali NetHunter VM	Via Gateway	Kali Linux	Offensive security and penetration testing toolkit
Arch Linux VM	Via Gateway	Arch Linux	General purpose Linux — secure comms, PQC operations, steganography
Ubuntu Dev VM	Via Gateway	Ubuntu 24.04 LTS	Development environment, AI App Builder pipeline
Vault VM	None (air-gapped)	Minimal Linux	Air-gapped secret and key storage — zero network interface
Disposable VM	Via Gateway	Ephemeral Debian	One-time-use throwaway — fresh per session, auto-wipe on close

2.3 VM Features

Feature	Implementation
Per-VM firewalls	12 toggleable nftables policy controls per VM — default-deny between all domains
Per-VM encrypted storage volumes	Independent encryption keys per VM — compromise of one VM yields no other VM data
Per-VM network identities	Different Tor circuits per VM — separate exit nodes and fingerprints
VM snapshots	Rollback after attack or corruption — clean state restoration from Dom0
Live VM migration	In cluster mode: migrate running VMs across physical devices over encrypted mesh
GPU isolation	Vulkan passthrough with per-VM GPU slices — no cross-VM GPU memory access
Hardware-accelerated virtualization	ARM64 VHE extensions — full hardware support; near-native performance overhead

SECTION 3 — NETWORK STACK & PRIVACY

Mandatory Anonymization — Multi-Layer DPI Defeat — Anti-Surveillance

All outbound traffic from user VMs is routed through the **Gateway VM**, which enforces mandatory anonymization. There is no path for a user VM to bypass the gateway — this is enforced at both the software (nftables) and hardware (HNCP on-wire) levels.

3.1 Core Network Components

Component	File	Function
Network Manager	network/network_manager.py	Routing policies, rules, and failover orchestration
Tor Controller	network/tor/	Tor service, bridge modes, circuit management, multi-hop routing
I2P Gateway	network/i2p/	I2P anonymous overlay network for I2P-native services
DNSCrypt	network/dnscrypt/	Encrypted DNS resolver — all resolution inside Gateway VM
VPN Controller	network/vpn/	Multi-VPN orchestration — optional additional egress routing
Firewall	network/firewall/	nftables rules engine — per-VM default-deny policies

3.2 Network Privacy Enhancement (v3.1.0 — Phase 18)

Component	File	Function
WebTunnel Transport	webtunnel_transport.py	Disguises Tor as HTTPS to Cloudflare/Fastly/Akamai CDN — defeats Deep Packet Inspection
V2Ray Transport	v2ray_transport.py	Multi-protocol support: WebSocket, HTTP/2, gRPC, QUIC
Behavioral Obfuscation	behavioral_obfuscation.py	ML-driven typing, browsing, and mouse movement pattern randomization
Phantom Activity	phantom_activity.py	Overlays decoy network requests to mask real activity patterns from traffic analysis
Bridge Manager	bridge_manager.py	Manages pluggable transports (obfs4, meek, Snowflake) for censorship circumvention
Traffic Shaper	traffic_shaper.py	Packet timing and size normalization — defeats timing correlation attacks
Transport Manager	transport_manager.py	Coordinates all transports; failover logic for resilient connectivity

3.3 Anti-Surveillance Network Features

Feature	Implementation
Ultrasonic detection	18–22 kHz bandpass circuit monitors for ultrasonic tracking beacons; automated speaker jamming response
Baseband isolation	Cellular modem accessible only via Gateway VM — baseband attack surface dramatically reduced
Zero DNS leaks	DNSCrypt handles all resolution inside Gateway; no DNS queries bypass anonymization
Kill switches	Dual hardware+software: immediate network termination on demand, no software bypass
Tor kill switch	Blocks all traffic if Tor circuit fails — no clearnet fallback under any failure mode
JA3/JA4 mimicry	TLS handshake fingerprints match legitimate browsers — defeats surveillance fingerprinting

SECTION 4 — CRYPTOGRAPHIC STACK

Full-Stack Post-Quantum Cryptography — No Standalone ECC

QWAMOS implements **defense-in-depth hybrid cryptography** — combining multiple post-quantum algorithms so that breaking any one does not compromise security. No standalone ECC is used anywhere in the security-critical path. All classical components are supplemented, never replaced, by post-quantum algorithms.

4.1 Key Encapsulation Mechanisms (KEMs)

Algorithm	Level	Standard	File
ML-KEM-1024 (Kyber-1024)	Level 5	NIST FIPS 203	crypto/kyber/
BIKE (Level 5)	Level 5	NIST Round 4 Alternate	crypto/pqc_next_gen/bike_kem.py
HQC (Level 5)	Level 5	NIST Round 4 Alternate	crypto/pqc_next_gen/hqc_kem.py
Classic McEliece (Level 5)	Level 5	NIST Round 4 Alternate	crypto/pqc_next_gen/mceliece_kem.py

Hybrid Construction: Hybrid KEM = Kyber-1024 XOR BIKE XOR HQC XOR McEliece XOR X25519. Breaking any single algorithm does not compromise the hybrid.

4.2 Digital Signature Schemes

Algorithm	Standard	Use
Dilithium (ML-DSA-87)	NIST FIPS 204	Primary signing; hardware-backed in HSM; storage key signatures; firmware attestation
Falcon-1024	NIST FIPS 206	Compact signatures — certificate-constrained environments
SPHINCS+-SHA2-256	NIST FIPS 205	Hash-based stateless backup — immune to all algebraic attacks

4.3 Symmetric Primitives

Algorithm	Role	Quantum Resistance
ChaCha20-Poly1305	Authenticated encryption (AEAD) — all transport and storage	Classical-secure (256-bit key)
BLAKE3	Hashing — integrity verification, key derivation input	Quantum-resistant at 256-bit output
Argon2id	Key derivation — GPU/ASIC-resistant KDF from passphrase	Memory-hard; resistant to dedicated hardware attacks
HKDF-BLAKE2b	Key expansion — derives per-purpose keys from master key material	Classical-secure; quantum-safe at 256-bit

4.4 Hardware Security Integration

Component	File	Function
Hardware Keystore	crypto/hardware_keystore.py	StrongBox/TEE abstraction layer
PQC Keystore	crypto/pqc_keystore.py	Hardware-backed post-quantum key storage
HSM Manager	crypto/hsm_manager.py	TPM 2.0 abstraction layer — Infineon SLB9672 integration
FIDO2 Backend	crypto/fido2_backend.py	WebAuthn/FIDO2/YubiKey hardware security key support
Key Rotation	crypto/key_rotation.py	Automatic scheduled key rotation — no long-lived static keys
BLAKE3 Hasher	crypto/blake3_hasher.py	Quantum-resistant hashing service across all modules
Argon2id KDF	crypto/argon2id_kdf.py	Memory-hard password derivation — GPU/ASIC resistant

Component	File	Function
Glass Photonic QRNG (v2)	crypto/qrng_driver.py (planned)	Quantum vacuum fluctuation entropy source — borosilicate glass chip, femtosecond laser direct writing. 42.7 Gbit/s demonstrated. CMRR >73 dB. 8-hour field stability. NIST SP 800-90B certification target. Replaces classical TRNG in v2 VALKYRJA hardware.

4.5 QKD Simulation (Phase 27)

Protocol	File	Description
BB84	crypto/pqc_next_gen/qkd_simulator.py	Bennett-Brassard 1984 — original prepare-and-measure QKD protocol
E91	crypto/pqc_next_gen/qkd_simulator.py	Ekert entanglement-based protocol — quantum nonlocality
Decoy State	crypto/pqc_next_gen/qkd_simulator.py	Practical fiber-optic QKD defense model

4.6 Glass Photonic Architecture — v2 Hardware Integration

The v2 VALKYRJA hardware platform integrates a borosilicate glass photonic chip as the primary hardware entropy source, replacing the classical TRNG. The same glass photonic platform supports both QRNG and CV-QKD on a single chip:

Function	Demonstrated Performance	Certification Target
Quantum RNG (QRNG)	42.7 Gbit/s secure random bits; CMRR >73 dB; 8-hour field stability without recalibration; ~1 dB insertion loss	NIST SP 800-90B → FIPS 140-3
CV-QKD Receiver	3.2 Mbit/s secret key rate over 9.3 km fiber; QPSK modulation; source-device-independent	ETSI QKD standards — v2 roadmap

Fabrication: femtosecond laser direct writing (FLDW) into Corning EAGLE XG borosilicate glass. No semiconductor cleanroom required. Glass is naturally polarization-insensitive, chemically inert, and thermally stable — quantum entropy properties are fundamental to the material.

SECTION 5 — AI & MACHINE LEARNING

Triple-AI Coordination — Real-Time Threat Detection — AI Governor

5.1 Triple-AI Coordination System

QWAMOS operates three AI models in concert, with a consensus engine adjudicating disagreements. All external AI calls are routed through Tor, sanitized by the Request Sanitizer, and executed inside an isolated AI Sandbox.

AI Model	Role	Access Mode
Kali GPT	Security analysis, threat assessment, exploit evaluation	On-device (local NPU inference)
Claude	Architecture design, code review, compliance guidance	Via Tor API (isolated channel)
ChatGPT	Code quality, UI/UX assistance, test generation	Via Tor API (isolated channel)

5.2 AI Security Infrastructure (Phase 20)

Component	File	Function
Multi-Model Orchestrator	ai_security/orchestrator.py	Coordinates all three AI models; majority-rule consensus synthesis
NL Interface	ai_security/nl_interface.py	Natural language security query interface
LoRA Fine-tuning	ai_security/lora_finetuning.py	On-device model fine-tuning for threat intelligence adaptation
Threat Intel	ai_security/threat_intel.py	Threat intelligence aggregation and classification
Report Generator	ai_security/report_generator.py	Automated security analysis and incident reports

5.3 ML Threat Detection (Phase 7)

Three independent machine learning models monitor for anomalies in real time from Dom0, monitoring all VM activity, network traffic patterns, and system calls.

Model	Type	Detects
Autoencoder	Unsupervised neural network (Dom0)	Novel/zero-day behavioral anomalies — unknown attack patterns
Random Forest	Supervised ensemble (Dom0)	Known attack signatures and patterns — database-backed classification
LSTM	Recurrent neural network (Dom0)	Temporal sequence attacks across time windows — APT detection

5.4 AI Governor (Phase 15)

The **hypervisor/ai_governor.py** module uses ML to make real-time resource allocation decisions across all VM domains: predictive CPU/memory scaling per VM, I/O priority scheduling based on workload prediction, and battery-aware resource balancing.

5.5 Behavioral Security

Feature	Module	Description
AI Typing Verification	keyboard/	Behavioral biometric profile of authorized user's typing rhythm — detects unauthorized user
ML Behavioral Obfuscation	network_privacy/behavioral_obfuscation.py	ML models add noise to typing, mouse, and browsing patterns to defeat behavioral fingerprinting

SECTION 6 — SECURITY MODULES

Emergency Protection — Anti-Surveillance — Firmware Security

6.1 Emergency Protection

Feature	Mechanism
Panic Wipe	Triple-tap 5-finger gesture instantly destroys all encryption keys and sensitive data — <2 seconds execution
Duress Profile	Special duress PIN unlocks convincing decoy profile, hiding real data — primary VMs remain encrypted
Network Kill Switch	Instantly terminates all network connectivity — software and hardware paths both severed
Tor Kill Switch	Blocks all traffic if Tor circuit fails — no clearnet fallback under any failure mode
VM Rollback	Restore any VM from snapshot after compromise — clean state from Dom0 command

6.2 Anti-Surveillance Features

Feature	Module	Mechanism
Ultrasonic Guard	security/ultrasonic_guard/	Detects 18–22 kHz ultrasonic tracking beacons; automated speaker jamming
Pegasus/Graphite Guard	security/pegasus_graphite_guard/	Mitigates NSO Group zero-click exploit vectors — baseband isolation + hypervisor memory protection
Ghost Module	security/ghost/	Comprehensive anti-surveillance feature suite including environmental sensors
Anti-Keylogging	keyboard/	Touch coordinate obfuscation (±5px random noise) defeats stylometric fingerprinting
Anti-Screenshot	UI layer	Android FLAG_SECURE blocks all screenshots and recordings across all profiles
Shoulder-Surfing Defense	keyboard/	Random keyboard layout reshuffled every 30 seconds against visual observation
TLS Fingerprint Masking	network_privacy/	JA3/JA4 evasion via fingerprint mimicry — traffic matches legitimate browsers

6.3 Firmware & Bootloader Security

Feature	File	Function
Firmware Integrity Monitor	security/firmware_integrity_monitor.py	Continuous bootloader and TrustZone integrity verification
ML Bootloader Override	security/ml_bootloader_override.py	ML model detects unauthorized firmware modifications — defeats evil maid attacks
A/B Partition Isolation	security/ab_partition_isolation.py	Isolates OS partitions; detects tampering between boot cycles
Vault 7 Mitigation	Integrated	Detects Dark Matter-style firmware persistence — Equation Group techniques mitigated

6.4 Advanced Security Suites

Module	Directory	Purpose
Aegis	security/aegis/	Advanced multi-vector threat detection and alerting — autonomous response
Chimera	security/chimera/	Multi-vector simultaneous attack response — adaptive countermeasures
RASP	qwamos_app_security/	Runtime Application Self-Protection — monitors app behavior in real time

SECTION 7 — SECURETYPE KEYBOARD

Per-Keystroke PQC Encryption — Layout Obfuscation — Anti-Surveillance

SecureType is a fully custom, security-hardened software keyboard implemented in **TypeScript / React Native**. It applies post-quantum encryption to every individual keypress before the keystroke leaves the keyboard subsystem — defeating all keyboard-level interception, including stylometric analysis and shoulder surfing.

Feature	Description
Per-Keystroke PQC Encryption	Every individual keypress encrypted with ML-KEM-1024 (Kyber) before leaving keyboard layer — makes raw keystroke capture worthless
Touch Coordinate Obfuscation	±5px random noise added to all touch coordinates — defeats stylometric fingerprinting even if coordinate data is exfiltrated
Random Layout Shuffle	Keyboard layout randomized every 30 seconds — memorized shoulder-surfing layout becomes useless
Anti-Screenshot	FLAG_SECURE prevents any screen capture while keyboard is active — any recording returns black frame
Clipboard Isolation	Clipboard contents encrypted and scoped per-VM — no cross-VM clipboard data access under any condition
AI Typing Verification	ML behavioral biometric detects unauthorized users by typing rhythm anomaly — can lock or alert

Directory	Contents
keyboard/src/	React Native UI components — custom keyboard layout engine
keyboard/crypto/	Keystroke-level ML-KEM-1024 encryption routines
keyboard/services/	Keyboard management services — layout scheduling, behavior monitoring
keyboard/config/	Encryption and behavior configuration — threshold tuning
keyboard/docs/	Technical documentation for keyboard security model

SECTION 8 — STORAGE & ENCRYPTION

Multi-Layer PQC Encryption — Per-VM Volumes — Vault VM

8.1 Encryption Stack

Layer	Algorithm	Function
Volume encryption	ChaCha20-Poly1305	Per-VM disk encryption — authenticated encryption with associated data
Key derivation	Argon2id	GPU/ASIC-resistant KDF from passphrase — memory-hard
Key storage	ML-DSA-87 + StrongBox	Hardware-backed PQC key storage in secure enclave
Secure deletion	3-pass DoD wipe	DoD-grade file deletion — key destruction + overwrite on VM teardown

8.2 Storage Components

Component	File	Function
PQC Volume	storage/pqc_volume.py	Post-quantum encrypted disk volumes — primary storage security layer
Volume Manager	storage/	LUKS/VeraCrypt wrapper and orchestration — unified volume management
Volume Snapshots	storage/volume_snapshots.py	VM disk snapshotting for rollback and forensic preservation
VeraCrypt Wrapper	storage/veracrypt/	VeraCrypt volume management for compatibility operations

8.3 Vault VM — Air-Gapped Storage

The **Vault VM** has zero network interface. It is used exclusively for:
- Long-term secret storage and key management
- Key import/export operations between encrypted volumes
- Air-gapped cryptographic operations (signing, verification)
- Password manager data — no network path for exfiltration

On VALKYRJA hardware, the Vault VM is assigned to the dedicated NVMe SSD volume (512 GB, air-gapped from all network VM storage paths at the storage controller level).

SECTION 9 — SECURE COMMUNICATIONS

Signal Protocol — ZRTP Voice — Steganography — Onion Dead Drop

Component	File	Protocol
Signal Client	qwamos_comms/signal_client.py	Signal Protocol — E2E encrypted messaging with forward secrecy
ZRTP Client	qwamos_comms/zrtp_client.py	ZRTP — Encrypted VoIP without PKI dependency; authentication via SAS
Sealed Sender	qwamos_comms/sealed_sender.py	Anonymous message delivery — hides sender metadata from infrastructure
Onion Mailbox	qwamos_comms/onion_mailbox.py	Tor-based dead drop for asynchronous communications
Disappearing Messages	qwamos_comms/disappearing_messages.py	Configurable auto-delete timers — key material destroyed on deletion
Message Queue	qwamos_comms/message_queue.py	Reliable message buffering for intermittent connectivity
Steganography	qwamos_comms/stego/	Covert message embedding in images/files for plausibly deniable transmission

9.1 Post-Quantum Transport

All communications use hybrid PQC key encapsulation: **ML-KEM-1024 + X25519 hybrid** for key exchange, **ML-DSA-87** for message authentication, and **ChaCha20-Poly1305** for authenticated encryption. This ensures quantum-resistant confidentiality while maintaining forward secrecy through ephemeral key generation.

SECTION 10 — AI APP BUILDER

Triple-AI Code Generation & Automated Security Auditing

The AI App Builder is a triple-AI automated code generation and security auditing pipeline. Users describe the desired application in natural language; the three AI models collaborate to design, implement, audit, and test the application inside the QWAMOS security sandbox.

10.1 Pipeline Flow

Step	Actor	Action
1	User	Describes desired app in natural language via NL Interface
2	Kali GPT	Assesses security requirements and identifies threat model
3	Claude	Designs architecture, code structure, and security controls
4	ChatGPT	Generates implementation code and UI components
5	Consensus Engine	Adjudicates conflicts between AI model recommendations
6	Security Auditor	Automatically scans all generated code for vulnerabilities
7	Test Generator	Creates unit and integration tests for the generated application
8	Delivery	Code is delivered to the requesting VM in its encrypted volume

10.2 Components

Directory	Contents
ai_app_builder/pipeline/	Triple-AI coordination and consensus engine implementation
ai_app_builder/auditor/	Automated security auditing of generated code — Bandit, Gitleaks integration
ai_app_builder/generator/	Code generation templates, runners, and language-specific tooling
ai_app_builder/tests/	Automated test generation framework
ai_app_builder/config/	AI thresholds, coordination configuration, and consensus rules

SECTION 11 — IOT ECOSYSTEM

Device Management — Smart Home Integration — Wearables — Collective Defense

Component	File	Function
Device Discovery	qwamos_iot/discovery/	Network scanning and device identification
Device Fingerprinting	qwamos_iot/discovery/	OS/vendor/version detection per IoT device
Vulnerability Checker	qwamos_iot/security/	CVE database lookup per device — automatic risk assessment
Firmware Manager	qwamos_iot/security/	Device firmware update management within secure environment
Anomaly Detector	qwamos_iot/security/	Behavioral analysis of device traffic — ML-based anomaly detection
Privacy Policy Manager	qwamos_iot/	Per-device traffic restrictions and data collection controls
Wearable Bridge	qwamos_iot/wearables/	Smartwatch integration layer — biometric auth and proximity unlock
Multi-Device Sync	qwamos_iot/sync/	Policy synchronization and threat intelligence sharing across devices
Collective Defense	qwamos_iot/sync/	Shared threat intelligence mesh — collective detection and response

Smart Home Integration

Home Assistant integration for smart home control within the secure QWAMOS environment. MQTT, Zigbee, and Z-Wave protocol support enables comprehensive IoT device management while maintaining the security isolation model.

SECTION 12 — FORENSICS & COMPLIANCE

Chain of Custody — Incident Response — CIS/NIST/Common Criteria

12.1 Forensics Capabilities (Phase 23)

Component	Directory	Function
Memory Forensics	qwamos_forensics/memory_forensics/	Live RAM analysis and artifact extraction
Forensic Collector	qwamos_forensics/evidence_collection/	Structured evidence gathering with chain of custody
Chain of Custody	qwamos_forensics/evidence_collection/	Tamper-proof, cryptographically signed evidence logs
Incident Response Orchestrator	qwamos_forensics/incident_response/	Automated response to detected incidents
Collect Forensics Script	tools/collect-forensics.sh	One-command evidence collection utility

12.2 Compliance Frameworks (Phase 24)

Framework	File	Scope
CIS Benchmarks	qwamos_compliance/cis_auditor.py	CIS Level 1 & 2 mobile controls
NIST 800-53	qwamos_compliance/nist_800_53.py	Federal security and privacy controls
OWASP	qwamos_compliance/owasp_assessment.py	Web/mobile application security assessment
Common Criteria	qwamos_compliance/common_criteria.py	CC Target of Evaluation (TOE) framework
SLSA Level 3	tools/generate-sbom.sh	Supply chain provenance — reproducible builds with signed manifests

12.3 Policy as Code

QWAMOS implements security policies as code with **policy/schema.json**. Policies are declared in YAML/JSON, compiled via **tools/policy-compile.sh**, signed with ML-DSA-87, and verified at boot time before any VM is started. This provides cryptographically verifiable policy enforcement with a complete audit trail.

SECTION 13 — CLUSTER MODE & FLUTTER UI SYSTEM

Multi-Device Mesh — Live VM Migration — Security Profiles

13.1 Cluster Mode (Phase 16)

Component	File	Function
Cluster Coordinator	cluster/cluster_coordinator.py	Manages multi-device mesh topology and device enrollment
Cluster Node	cluster/cluster_node.py	Per-device cluster participation and resource sharing

Feature	Description
Live VM migration	Migrate running VMs across physical devices over encrypted mesh — zero downtime
Shared resource pool	Distribute CPU, memory, and storage across cluster nodes for high-demand operations
Encrypted mesh networking	All cluster communication encrypted with ML-KEM-1024 + ChaCha20-Poly1305
Collective threat intelligence	Threat detections shared across cluster — early warning for all nodes

13.2 Flutter UI System (Phase 11)

The QWAMOS user interface is implemented in **Flutter (Dart)** with 26 custom widgets, GLSL shader-based visual effects, and Material Design 3 theming. The complexity of the hypervisor is completely abstracted through the four-profile security model.

Profile	VM Configuration	FMP State	Use Case
Everyday	Android VM active; standard Tor circuit	Off	Normal daily use — maximum UX comfort
Private	Fresh isolated Android VM; separate Tor identity	Standard Privacy	Sensitive browsing, transactions, messaging
Secure	Arch Linux VM; PQC comms; ZRTP voice; full E2E stack	Maximum Privacy	Classified comms, legal-sensitive, financial
Ghost	Fresh Disposable VM; WebTunnel DPI defeat; Phantom Activity overlay	Maximum Privacy	Hostile environment, surveillance counter-ops

SECTION 14 — ONYX HARDWARE INTEGRATION

VALKYRJA Device — Required Software Modifications

QWAMOS v3.2.0 (targeted for VALKYRJA device launch) requires the following hardware abstraction layer modifications and new development to support Onyx VALKYRJA hardware. These items are the primary software integration workstreams for the device launch.

Integration Item	Description	Priority
RK3588 HAL	Kernel drivers for RK3588 peripherals, camera ISP, audio codec, power management within QWAMOS Linux kernel 6.6 LTS base	P0 — Critical
Display controller integration	QWAMOS profile state → FMP hardware signal path; display controller driver for Samsung LEAD 2.0 panel with FMP mode switching API	P0 — Critical
HNCP firmware	RISC-V MCU firmware for Hardware Network Co-Processor; implements traffic isolation enforcement, Gateway VM watchdog, and kill switch protocol	P0 — Critical
Tamper MCU firmware	Cortex-M0+ firmware for tamper detection; implements tamper condition classification, alert routing, and HSM key destruction trigger	P0 — Critical
HSM integration	Infineon SLB9672 driver and key management API integration with QWAMOS PQC keystore; ML-DSA-87 hardware-backed operations	P0 — Critical
Nuclear power rail management	Isolated power domain monitoring; HNCP and Tamper MCU power state management under nuclear cell	P1 — High
Solar MPPT integration	Perovskite QD solar controller driver; charge path management without cross-contaminating nuclear rail	P1 — High
VM allocation tuning	Increase Dom0/Guest VM memory ceilings to 32 GB LPDDR5X; update AI Governor CPU allocation for RK3588 8-core topology	P1 — High
UFS 4.0 + NVMe dual storage	Storage controller driver update; Vault VM assigned to NVMe; per-VM encrypted volume key management for dual-medium topology	P1 — High

SECTION 15 — BUILD SYSTEM & TESTING

Reproducible Builds — SLSA Level 3 — Comprehensive Test Suite

Build Script	repro-build.sh — reproducible build with deterministic output
SLSA Provenance	SLSA Level 3 — signed build manifests with documented build process
GPG Signing	All releases GPG-signed; signature verification required for installation
SBOM Generation	tools/generate-sbom.sh — Software Bill of Materials per release (EO 14028 compliance)
Policy Compilation	tools/policy-compile.sh — compiles and signs security policies
Forensics Collection	tools/collect-forensics.sh — one-command evidence collection

Test Suite

Test Category	Description
Unit tests	Per-module unit testing for all Python and TypeScript components
Integration tests	Cross-module integration testing with VM lifecycle coverage
KVM hardware tests	Tests requiring actual KVM hardware — boot, VM creation, isolation
Differential KVM/QEMU tests	Comparative tests between hardware KVM and QEMU emulation — regression detection
Security tests	Fuzzing, injection testing, policy enforcement validation
Hypervisor tests	VM isolation verification, cross-VM data leakage detection, side-channel testing

SECTION 16 — PHASE COMPLETION & ROADMAP

All 27 Phases Complete — v3.1.0 (January 2026)

Phase	Name	Status	Key Deliverables
1	Core Hypervisor	✓ Complete	ARM64 KVM hypervisor, VM lifecycle management
2	Network Isolation	✓ Complete	Whonix-style gateway, Tor/I2P routing
3	Storage Encryption	✓ Complete	Per-VM encrypted volumes, key management
4	Post-Quantum Crypto	✓ Complete	Kyber-1024, ChaCha20-Poly1305, BLAKE3, Argon2id
5	Firewall & Policies	✓ Complete	Per-VM firewalls, 12 policy toggles, panic wipe
6	AI Integration	✓ Complete	Kali GPT, Claude, ChatGPT triple-AI coordination
7	ML Threat Detection	✓ Complete	Autoencoder, Random Forest, LSTM detection models
8	SecureType Keyboard	✓ Complete	Per-keystroke PQC, coordinate obfuscation
9	AI App Builder	✓ Complete	Triple-AI code generation, automated security audit
10	Hardware Security	✓ Complete	ML bootloader lock, Vault 7 mitigation
11	Flutter UI	✓ Complete	26 custom widgets, GLSL shaders, Material Design 3
12	KVM Acceleration	✓ Complete	VirtIO drivers, hardware virtualization optimization
13	PQC Storage	✓ Complete	ML-DSA-87 hardware-backed keys, StrongBox integration
14	GPU Isolation	✓ Complete	Per-VM GPU slices, Vulkan passthrough
15	AI Governor	✓ Complete	ML resource allocation, predictive VM scaling
16	Cluster Mode	✓ Complete	Multi-device mesh, live VM migration
17	HSM Integration	✓ Complete	TPM 2.0, FIDO2, TrustZone integration
18	Network Privacy	✓ Complete	WebTunnel, V2Ray, behavioral obfuscation (v3.1.0)
19	Offensive Security	✓ Complete	Vulnerability scanner, penetration testing toolkit
20	Enhanced AI/LLM	✓ Complete	Multi-model orchestration, LoRA fine-tuning
21	App Security	✓ Complete	RASP, package verification, auditing
22	Secure Comms	✓ Complete	Signal Protocol, ZRTP, steganography
23	Forensics	✓ Complete	Memory forensics, chain of custody
24	Compliance	✓ Complete	CIS, NIST 800-53, OWASP, Common Criteria
25	UX/Accessibility	✓ Complete	React Native UI, voice control, gesture navigation
26	IoT Ecosystem	✓ Complete	Device discovery, MQTT/Zigbee/Z-Wave, wearables
27	Quantum Resistance	✓ Complete	BIKE, HQC, McEliece, QKD simulation, hybrid crypto

Roadmap

Timeframe	Planned Features
Near-Term (2026)	Satellite communication support (Starlink / Iridium); hardware wallet integration (Ledger, Trezor); post-quantum encrypted voice calls (full stack); additional HSM vendor support
Medium-Term (2027)	Desktop x86_64 port (Qubes OS compatibility layer); tablet form factor optimization; Enterprise MDM API; SLSA Level 3+ formal certification; secure OTA update infrastructure
Long-Term (2028+)	Hardware pre-installation partnerships; real QKD integration (fiber QKD hardware); decentralized identity framework; fully autonomous peer-to-peer mesh networking; formal verification of hypervisor isolation

QWAMOS vs. Alternatives — Security Capability Comparison

Capability	QWAMOS	GrapheneOS	CalyxOS	Standard Android
Full VM isolation (hypervisor)	YES	No (sandbox)	No (sandbox)	No
Mandatory Tor/I2P for all traffic	YES	Optional VPN	Optional VPN	No
Post-quantum cryptography	YES — full stack	No	No	No
QKD simulation protocols	YES	No	No	No
AI threat detection (3 ML models)	YES	No	No	No
Triple-AI app builder	YES	No	No	No
Ultrasonic detection & jamming	YES	No	No	No
Pegasus/Graphite zero-click mitigation	YES	Partial	No	No
Behavioral fingerprint obfuscation	YES	No	No	No
DPI-defeating WebTunnel transport	YES	No	No	No
Per-keystroke PQC encryption	YES	No	No	No
Live VM migration (multi-device)	YES	No	No	No
IoT security ecosystem	YES	No	No	No
Reproducible builds + SLSA	YES	Yes	Partial	No
Collective defense mesh	YES	No	No	No

QWAMOS v3.1.0 — Technical Specification
Qubes+Whonix Advanced Mobile Operating System
Obsidian Circuit | Desirae Stark, Chief Architect | April 2026
AGPL-3.0 | github.com/Dezirae-Stark/QWAMOS | qwamos@tutanota.com