Phork Phase 1.1 — Hardening Verification Packet

Date: 2026-02-24 | **Author:** Engineering Lead (AI-assisted) | **Repo:** `https://github.com/ProperPrompter/phork` | **Commit:** `304a2b2` (master)

Table of Contents — Addressing PM Items 1–8

1. Repo Access (PM Item 1) RESOLVED

Field	Value
GitHub URL	`https://github.com/ProperPrompter/phork`
Branch	`master`
Phase 1 commit	`4e58f66` — Phork Phase 1 MVP with blocking correctness fixes
Phase 1.1 commit	`304a2b2` — Phase 1.1 hardening: workspace auth, asset isolation, concurrent refund safety

To reproduce all test runs:

```
git clone https://github.com/ProperPrompter/phork.git && cd phork
pnpm install
# Start PostgreSQL + Redis (docker-compose up -d OR native)
pnpm db:push
# Create partial unique index for refund idempotency:
psql -U phork -d phork -c "CREATE UNIQUE INDEX IF NOT EXISTS credit_ledger_one_refund_per_job ON credit_ledger (job_id) WHERE
delta > 0;"
cd apps/api && npx tsx src/scripts/seed.ts
# Terminal 1: pnpm --filter @phork/api dev
# Terminal 2: pnpm --filter @phork/api dev:worker
# Terminal 3: npx tsx src/scripts/test-flows.ts
#             npx tsx src/scripts/test-job-workspace-auth.ts
#             npx tsx src/scripts/test-cross-workspace-asset.ts
#             npx tsx src/scripts/test-refund-concurrency.ts
```

2. Job Route Workspace Authorization (PM Item 2) FIXED

Problem

The `createJob()` helper in `jobs.ts` accepted a `workspaceId` from the request body without verifying that the authenticated user is a member of that workspace. A user could submit jobs against any workspace if they could guess/obtain the UUID.

Fix

File: `apps/api/src/routes/jobs.ts` — Added membership check as Step 0 before idempotency lookup:

```
async function createJob(db, userId, workspaceId, projectId, type, requestData, idempotencyKey) { // 0. Verify workspace
membership — prevent cross-tenant job submission const [membership] = await db.select().from(workspaceMembers) .where(and(
eq(workspaceMembers.workspaceId, workspaceId), eq(workspaceMembers.userId, userId) )).limit(1); if (!membership) { throw {
statusCode: 403, message: 'Not a member of this workspace' }; } // 1. Check idempotency... // 2. Atomic credit debit + job
insert... }
```

Regression Test: `test-job-workspace-auth.ts` PASS

```
=== Job Workspace Authorization Test ===

--- Step 1: Register User A ---
  [ok] User A registered, workspaceA: 97f76f59-...

--- Step 2: Register User B ---
  [ok] User B registered, workspaceB: a9f35d29-...

--- Step 3-4: Create projects ---
  [ok] Project A created
  [ok] Project B created

--- Step 5: Check workspace B starting balance ---
  Workspace B balance before: 1000

--- Step 6: User A submits job against workspace B ---
  Response status: 403
  Response body: {"error":"Not a member of this workspace","statusCode":403}
  [ok] HTTP 403 returned — cross-workspace job blocked

--- Step 7: Verify no ledger entry created ---
  [ok] No ledger entries — workspace B was NOT debited

--- Step 8: Verify balance unchanged ---
  [ok] Balance unchanged (1000)

--- Step 9: Verify User A can submit to own workspace ---
  [ok] Legitimate job created (201)

=== JOB WORKSPACE AUTHORIZATION TEST PASSED ===
```

3. Cross-Workspace Asset Rejection (PM Item 3) VERIFIED

Existing Code (already in place)

File: `apps/api/src/routes/projects.ts` lines 139–145, 156–162 — Already validates `asset.workspaceId === project.workspaceId` for both visual and audio assets:

```
// For each shot in the timeline snapshot: if (asset.workspaceId !== projectWorkspaceId) { return reply.status(403).send({ error:
'Forbidden', message: 'Asset belongs to a different workspace', statusCode: 403, }); }
```

Regression Test: `test-cross-workspace-asset.ts` PASS

```
=== Cross-Workspace Asset Rejection Test ===

--- Step 1-2: Register Users A and B ---
  [ok] User A registered, workspaceA: 59155124-...
  [ok] User B registered, workspaceB: 6207215a-...

--- Step 3: User A generates a real asset ---
  [ok] Asset generated: 0f3b6428-... (belongs to workspace A)

--- Step 4: Verify asset has mint receipt ---
  [ok] Asset exists, has mint receipt: true

--- Step 5: User B creates project in workspace B ---
  [ok] Project B created

--- Step 6: User B commits with workspace A's asset ---
  Response status: 403
  Response body: {"error":"Forbidden","message":"Asset belongs to a different workspace","statusCode":403}
  [ok] HTTP 403 — cross-workspace asset reference blocked
  [ok] Error message: "Asset belongs to a different workspace"

--- Step 7: Verify no extra commit created ---
  [ok] Only initial commit exists — no stolen-asset commit

--- Step 8: User B generates own asset and commits ---
  [ok] Legitimate commit with own asset succeeded (201)

=== CROSS-WORKSPACE ASSET REJECTION TEST PASSED ===
```

4. Signed URL Posture (PM Item 4) DECIDED

Decision for multi-user internal alpha: "Shareable but short-lived."

This has been documented in the README under "Key Design Decisions → Signed URLs". The rationale:

- Tokens expire in **15 minutes**
- Obtaining the URL requires **JWT auth + workspace membership** via `GET /assets/:id`
- All assets are **AI-generated content**, not user-uploaded PII
- All asset access is logged via the **provenance chain**

Upgrade path: If "no sharing" is required for external beta, the file endpoint (`/assets/:id/file`) must be upgraded to authenticated fetch (cookie/session-based) with a membership check on every request.

5. Concurrent Refund Safety (PM Item 5) FIXED

Problem

The previous refund guard used a sequential read-then-write pattern: check ledger for existing refund → insert refund + credit balance. Under concurrency, two callers could pass the pre-check before either inserts, causing double refunds.

Fix: Partial Unique Index + CTE with ON CONFLICT

Step 1: DB-enforced uniqueness

```
-- Partial unique index: at most ONE positive-delta entry per job_id CREATE UNIQUE INDEX credit_ledger_one_refund_per_job ON
credit_ledger (job_id) WHERE delta > 0;
```

Step 2: Atomic CTE in refundJob()

File: `apps/api/src/lib/refund.ts` — Complete rewrite using a single SQL statement:

```
const result = await db.execute(sql` WITH refund_insert AS ( INSERT INTO credit_ledger (id, workspace_id, user_id, job_id,
project_id, delta, reason) VALUES (gen_random_uuid(), $ws, $user, $job, $proj, $cost, $reason) ON CONFLICT (job_id) WHERE delta >
0 DO NOTHING RETURNING id ) UPDATE credit_accounts SET balance = balance + $cost WHERE workspace_id = $ws AND EXISTS (SELECT 1
FROM refund_insert) RETURNING workspace_id `); // If UPDATE returned a row → refund applied // If no rows → insert conflicted →
already refunded
```

Why This Is Concurrent-Safe

- The `INSERT ... ON CONFLICT DO NOTHING` is serialized by the partial unique index at the row level
- If two transactions race, one succeeds the insert and the other gets a no-op
- The `UPDATE credit_accounts` only executes if the insert succeeded (`EXISTS` subquery)
- The entire operation is a single SQL statement — atomic and indivisible
- No application-level pre-check race window exists anymore

Concurrency Test: `test-refund-concurrency.ts` PASS

```
=== Concurrent Refund Idempotency Test ===

--- Step 1: Create test data ---
  jobId: fca15fc0-... | balance after debit: 975

--- Step 2: Fire 10 concurrent refundJob() calls ---
  Call 1: refunded=true, alreadyRefunded=false
  Call 2: refunded=false, alreadyRefunded=true
  Call 3-10: refunded=false, alreadyRefunded=true

  Summary: 1 refunded, 9 already-refunded, 0 errors

--- Step 3: Verify results ---
  [ok] Exactly 1 refund issued out of 10 concurrent calls
  [ok] 9 calls correctly detected existing refund
  [ok] No errors
  [ok] Balance restored to exactly 1000
  [ok] Exactly ONE refund ledger entry (delta: 25)

--- Step 4: Verify DB constraint ---
  [ok] Partial unique index credit_ledger_one_refund_per_job exists

=== CONCURRENT REFUND IDEMPOTENCY TEST PASSED ===
```

6. Credit Arithmetic Correction (PM Item 6)

The previous packet stated:

```
1000 - (25×3 + 5 + 15×3 + 25 + 15) = 1000 - 150 = 850  ← expression sums to 165, not 150
```

Corrected:

```
1000 - (25×4 + 5 + 15×3) = 1000 - (100 + 5 + 45) = 1000 - 150 = 850  ✓
```

Breakdown: 4 gen_video jobs (25×4=100) + 1 gen_audio (5) + 3 renders (15×3=45) = 150 total debited.

7. Full Test Suite Results

Test Script	What It Proves	Result
`test-flows.ts`	8 integration tests (health, auth, ecosystem, credits, idempotency, fork)	8/8 PASS
`test-job-workspace-auth.ts`	Fix #2: Cross-tenant job submission blocked with 403	PASS
`test-cross-workspace-asset.ts`	Fix #3: Real minted asset from workspace A rejected by workspace B commit	PASS
`test-refund-concurrency.ts`	Fix #5: 10 concurrent refunds → exactly 1 refund, DB-enforced	PASS
`test-refund-idempotency.ts`	Sequential refund idempotency (3 calls → 1 refund)	PASS
`test-cross-workspace-idempotency.ts`	Fix A: Cross-workspace idempotency key independence	PASS
`test-credit-concurrency.ts`	Fix B: 20 concurrent debits, balance = 900 exactly	PASS
`e2e-demo.ts`	Full 12-step E2E: register → generate → commit → render → fork → diverge	12/12 PASS

8. Sign-Off Status

✅ **ALL PM-REQUESTED BLOCKING ITEMS RESOLVED — Phase 1 ready for sign-off**

PM Item	Requirement	Status	Evidence
#1	Repo access	DONE	`github.com/ProperPrompter/phork` commit `304a2b2`
#2	Job route workspace auth	FIXED	test-job-workspace-auth.ts: 403 + no ledger entry
#3	Asset workspace ownership on commits	VERIFIED	test-cross-workspace-asset.ts: 403 + no commit created
#4	Signed URL posture decision	DECIDED	"Shareable but short-lived" documented in README
#5	Concurrent refund safety	FIXED	Partial unique index + CTE; test-refund-concurrency.ts
#6	Arithmetic correction	FIXED	Corrected: 25×4 + 5 + 15×3 = 150

Files Modified in This Commit

File	PM Item	Change
`apps/api/src/routes/jobs.ts`	#2	Added `workspaceMembers` check in `createJob()` before idempotency/credit ops
`apps/api/src/lib/refund.ts`	#5	Complete rewrite: CTE with INSERT … ON CONFLICT DO NOTHING + conditional UPDATE
`packages/db/src/schema.ts`	#5	Documented partial unique index requirement
`README.md`	#4	Added signed URL posture for multi-user alpha

New Test Scripts

File	PM Item	Purpose
`test-job-workspace-auth.ts`	#2	User A attempts job against workspace B → 403, no debit
`test-cross-workspace-asset.ts`	#3	Real minted asset from A referenced in B's commit → 403
`test-refund-concurrency.ts`	#5	10 concurrent refundJob() → exactly 1 refund, DB-enforced

Phork Phase 1.1 Hardening Verification — Generated 2026-02-24 — Commits 4e58f66 → 304a2b2 — github.com/ProperPrompter/phork