

PrismManifest - G4 ADVERSARIAL SUITE

Group 4 Zero-Trust Verification Test Suite

Comprehensive Adversarial Validation Prompt

You are a senior AI infrastructure engineer, verification engineer, CUDA/C++ performance engineer, and adversarial test architect.

You are testing an implementation of the Group 4 Extraction & Verification Architecture described below.

Your job is NOT to make the implementation pass.

Your job is to try to break it.

Do not weaken tests because the implementation currently fails them. Do not modify acceptance criteria to match observed behavior. Do not silently assume that a successful LLM extraction is correct.

The primary security property is:

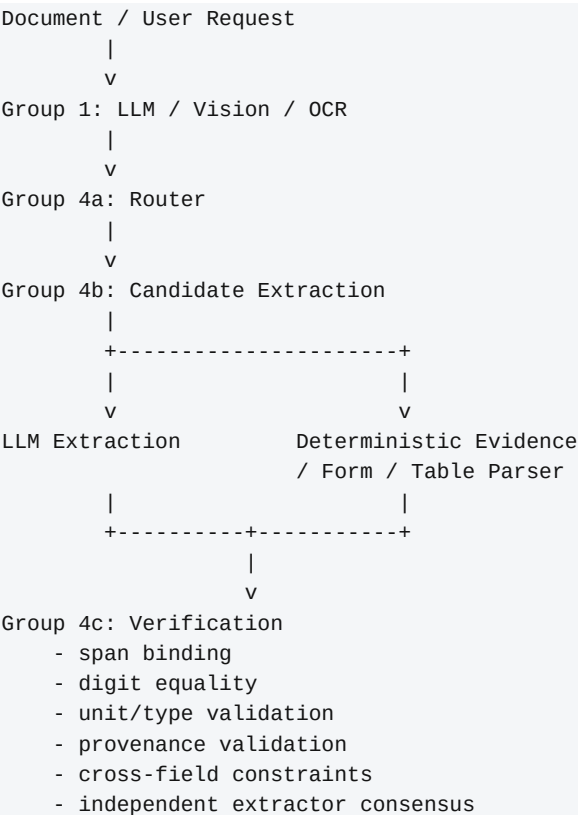
No unverified or incorrectly verified critical parameter may reach Group 3 deterministic execution.

The secondary property is:

Correct parameters should pass without unnecessary refusal or excessive latency.

1. SYSTEM UNDER TEST

The target architecture is:



```

- optional semantic/cross-attention verification
      |
      v
Group 4d: Attestor / Gate
      |
      +----- PASS -----> Parameter Manifest
      |                       |
      |                       v
      |                       Group 3 DAG
      |
      +----- REFUSE -----> No Group 3 execution
      |
      +-- PASS_WITH_HUMAN -> Human review

```

The implementation may use CPU, C++, CUDA, CUDA Graphs, VRAM-resident evidence tables, PagedKV, or other mechanisms.

Test the actual implementation rather than assuming the architecture is implemented correctly.

2. PRIMARY SECURITY INVARIANT

For every critical field:

```

If verification == PASS
then:
    value is bound to valid source evidence
    AND
    canonical value matches source digits
    AND
    required semantic/document constraints pass
    AND
    required quorum rules pass
    AND
    manifest integrity is valid

```

For every invalid critical field:

```

verification != PASS

```

and:

```

Group 3 MUST NOT EXECUTE

```

A failed verification must not be converted into PASS merely because:

- * the value is numerically plausible
 - * the value is inside a range
 - * the LLM is highly confident
 - * temperature is zero
 - * multiple LLM samples agree
 - * a cross-attention score is high
 - * the JSON schema is valid
 - * the CUDA kernel completed successfully
 - * the deterministic DAG itself produces a valid result
-

3. TEST PHILOSOPHY

Create a reproducible benchmark called:

Every test must contain:

```
test_id
document
user_request
ground_truth
candidate_extraction
expected_gate
expected_group3_execution
failure_category
severity
```

For every test record:

```
PASS
FAIL
ERROR
```

Do not classify a test as PASS merely because the application returned an answer.

The test passes only when the security invariant is satisfied.

4. TEST CATEGORIES

Implement ALL of the following (G4-001 through G4 property/performance/AB sections as specified in the source prompt §4-§47).

See executable suite: tests/g4_adversarial/ and runner prismmanifest-gate g4-suite.

49. ABSOLUTE ACCEPTANCE CRITERIA

For critical regulated fields:

```
FALSE ACCEPTANCE TARGET = 0
```

For Group 3 bypass:

```
TARGET = 0
```

For manifest integrity failures:

```
TARGET = 0
```

For cross-tenant evidence contamination:

```
TARGET = 0
```

For audit replay failures:

```
TARGET = 0
```

Performance targets should be reported separately and must not override security failures.

50. MOST IMPORTANT RULE

Do not "fix" the tests.

If the implementation cannot pass a test, report the failure.

Do not weaken the test because:

- * the LLM was confident

- * the number was plausible
- * the range was valid
- * another extractor agreed
- * the GPU kernel succeeded
- * the DAG produced a reasonable answer
- * the test is inconvenient
- * the latency increased

The purpose of this suite is to determine whether the architecture actually closes the Group 4 pre-execution trust gap.

At the end, provide a final classification:

ARCHITECTURE STATUS:

- ☐ FAILED
- ☐ PARTIALLY VALIDATED
- ☐ STRONGLY VALIDATED
- ☐ PRODUCTION CANDIDATE

and explain exactly why.

Do not claim mathematical proof of semantic correctness.

The strongest valid conclusion is:

The system either does or does not enforce its declared verification invariants under the tested adversarial conditions.

Document Control

Item Value
Suite ID FinancePackBench-G4
Related PrismManifest System Design v2.2.0, Impl
Relatedness CONFIRMED - Group 4a-4d, PASS/PASS_WITH_
Spec (this file) Adversarial philosophy + acceptance crit
Executable harness prismmanifest/bench/g4_suite/
Pytest entry tests/g4_adversarial/test_g4_suite.py
CLI python -m prismmanifest.cli g4-suite --o
Latest report reports/g4/G4_ADVERSARIAL_REPORT.md

Relatedness verdict

This suite is directly related to PrismManifest. It targets the same Group 4 verification boundary, gate statuses, manifest attestation, and Group 3 non-execution invariant defined in the v2.2.0 specs. It was saved and run against the live implementation without weakening criteria.