**Blend Audit Summary (Certora) – By Oluwaseun David Olajide (web3 giant)**

During the Certora-based audit of the Blend project, I identified a series of technical inconsistencies and misconfigurations that could negatively impact compilation, public interface usability, and the long-term maintainability of the crate. This report categorizes and expands on the issues found, highlights their potential implications, and suggests practical solutions.

1. **Missing Dependencies**

**Summary:** Several crates (soroban\_env\_host, ed25519\_dalek, serde\_json, rand) are used directly in the source code but not declared in Cargo.toml. This causes immediate compilation failures for any developer trying to build the project.

**Locations:**

* soroban\_env\_host: src/env.rs:972, 1499, 1599
* ed25519\_dalek: src/testutils/sign.rs:22, 53, 63
* serde\_json: src/testutils.rs:70, 80, 102, 112
* rand: src/testutils.rs:428

**Implications:**

* Fails to compile out of the box, breaking CI/CD pipelines.
* Increases onboarding time for contributors unfamiliar with missing dependencies.
* Indicates a potential disconnect between development and build/test environments.

**Recommendation:**

Declare each missing crate in Cargo.toml under [dependencies] or [dev-dependencies], depending on context.

**2.Incorrect Module Paths**

**Summary:** Multiple internal modules such as internal::Host, LedgerInfo, Budget, etc., are referenced without being properly defined or exported.

**Locations:**

* src/env.rs: Lines 484, 508, 532, 535, 536
* src/ledger.rs:179
* src/testutils.rs: Lines 134, 135, 324, 339

**Implications:**

* Compilation errors due to unresolved paths.
* Broken abstraction boundaries—internal modules aren't correctly exposed or organized.
* Prevents external crates from integrating with public-facing APIs.

**Recommendation:**

Ensure these modules are defined and properly exposed in the crate’s lib file, or refactor to use a consistent module pathing strategy.

**3. panic\_impl Definition**

**Summary:** A duplicate implementation of the panic\_impl lang item was found, which conflicts with the one provided by the standard library.

**Location: src/lib.rs:64**

**Implications:**

* Causes linking errors in standard environments.
* Breaks builds in both no\_std and std targets.
* Potential confusion about panic behavior in constrained environments.

**Recommendation:**

Remove the custom panic\_impl unless specifically targeting no\_std and building a runtime crate that requires it.

**4. Incorrect Trait Implementation**

**Summary:** The crate defines a method escalate\_error\_to\_panic for the trait internal::EnvBase, which doesn’t actually exist in the trait's definition.

**Location: src/env.rs:1803**

**Implications:**

* Results in compiler trait implementation errors.
* Misleads future developers expecting this method to exist in EnvBase.
* Violates Rust's trait safety and coherence principles.

**Recommendation:**

# Either add the method to the trait definition or remove the implementation altogether.

**5. Missing Reexports in no\_std Mode**

**Summary:** Types like String, Vec, and Box are used, but not explicitly reimported from the alloc crate in no\_std environments.

**Locations:**

* src/env.rs:1665
* src/logs.rs:144
* src/testutils/arbitrary.rs:1117

**Implications:**

* Breaks builds in embedded or WASM targets where std is unavailable.
* Inhibits portability across platforms—a key promise of Rust's ecosystem.
* Obstructs future use of this crate in blockchain VMs or smart contract runtimes.

**Recommendation:**

Use use alloc::{vec::Vec, boxed::Box, string::String} to properly support no\_std.

**6.Public API Inconsistencies**

**Summary:** Public-facing interfaces reference types like Host and LedgerInfo that are not exported or do not exist.

**📍 Locations:**

* src/env.rs:484, 508, 532

**Implications:**

* Breaks consumers of the crate relying on public API.
* Introduces ambiguity in interface contracts.
* Hinders effective documentation and library adoption.

**Recommendation:**

Ensure all types in the public API are either fully exported or abstracted through well-defined interfaces.

**7.Feature Mismanagement**

**Summary:** Crates like serde, rand, etc., are used but not properly declared as optional features.

**Locations:**

* src/testutils.rs:70, 428

**Implications:**

* Feature flags are a core part of crate flexibility in Rust.
* Causes hard dependency bloat—users cannot opt out of unused dependencies.
* May introduce incompatibilities for downstream crates that already use serde or rand.

**Recommendation:**

Add [features] block in Cargo.toml, e.g.:

toml

CopyEdit

[features]

serde = ["serde", "serde\_json"]

rand = ["rand"]

**8. Documentation Gaps**

**Summary:** The crate’s documentation lacks any mention of required dependencies or optional features, leaving users in the dark during setup.

**Location: N/A (global)**

**Implications:**

* Reduces developer confidence and slows adoption.
* Forces users to "trial-and-error" their way through installation.
* Contradicts good documentation principles, especially for open-source or shared codebases.

## OTHER VULNERABILITIES SEEN

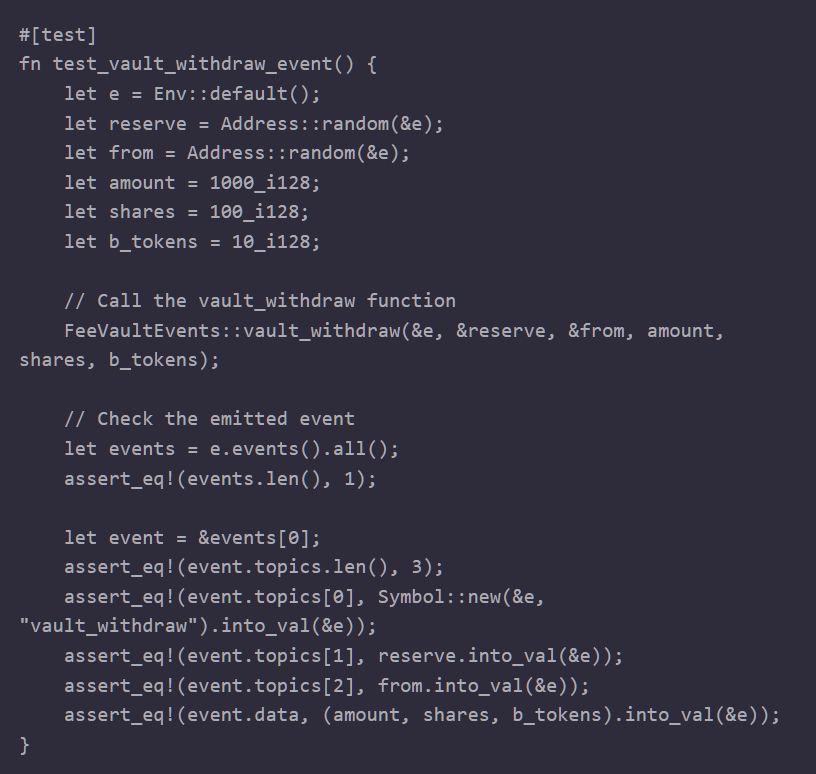
I ran cargo mutate on the vault code and I found a rather concerning vulnerability that could be a high or medium case this was how it was reported:

So all the test suites pass which is great but there is just one thing the test suites don't validate the withdrawal eventfrom the reserve vault and the amount being withdrawn from the reserve vault this is crucial to the security of the safety of the vault because anybody can just steal whatever amount they want and worse nothing to prove that the withdrawal happened.

Explain and rationalize the potential impact. Provide:

<https://github.com/code-423n4/2025-02-blend/blob/main/fee-vault/src/events.rs> line 48 in this event line if i change this Symbol::new(&e, "vault\_withdrawal"), to this Symbol::new(&e, "") there is no test for the withdrawal from the vault.

I mean just look at the result after running cargo test  
test result: ok. 42 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in50.49s. Below is a test written to check for withdrawal:



And below is another test written to check for amount being withdrawn:

