GranEverest — EverestVault Test & Analysis Report
Contracts test and review summary (Hardhat, Slither, Foundry fuzz & manual testing)

1. Environment & Scope

Tooling
- Hardhat: 2.26.x (TypeScript)
- Solidity compiler: 0.8.24 (EVM: paris)
- Foundry (forge): 1.4.4-stable
- Slither (static analysis) installed via pipx

Repo layout (contracts/)
- src/EverestVault.sol
- src/EverestVaultMulti.sol
- src/MockToken.sol
- src/GE_Timelock.sol
- src/LoopTester.sol
- foundry-test/EverestVaultMultiFuzz.t.sol

Foundry config (foundry.toml)
- src = src
- test = foundry-test
- libs = lib
- out = out-foundry
- cache_path = cache-foundry
- solc_version = 0.8.24
- evm_version = paris
- remappings:
  - @openzeppelin/contracts -> node_modules/@openzeppelin/contracts
  - forge-std -> lib/forge-std/src

Scope of this report: automated tests and manual reviews on the EverestVault / EverestVaultMulti codebase as of the generation date above.

2. Hardhat Test Suite Summary

Command
npx hardhat test

Result
17 passing tests.

Suites and checks

1) EverestVault — Anti-loop atomic (same block)
- Blocks a deposit if it occurs in the same block as a borrow.
- Allows deposit in a later block even when there is existing debt.

2) EverestVaultMulti

Deployment
- Sets feeRecipient correctly and rejects the zero address.

Vault creation
- Allows users to create vaults and tracks the vault count.

deposit
- Reverts if no ETH is sent or the vault does not exist.
- Adds collateral (net of the 0.25% protocol fee), updates totals and sends the fee.

borrow
- Reverts if amount == 0, the vault is missing or there is no collateral.
- Allows borrowing up to the max LTV (70%) and reverts above that limit.
- Updates lastBorrowBlock on borrow (anti-loop guard).

repay
- Reverts if no ETH is sent, the vault is missing or there is no debt.
- Repays the exact debt and clears it.
- Repays debt and refunds excess ETH when overpaying.

withdraw
- Reverts if amount == 0, the vault is missing or there is not enough collateral.
- Withdraws collateral when there is no debt, applies the 0.25% fee and updates totals.
- Reverts if the withdrawal would violate the max LTV with existing debt.

pause
- Only the owner can pause/unpause.
- When paused: deposit, borrow and withdraw revert; repay remains allowed.

3. Slither Static Analysis Summary

Tool
Slither (static analysis)

Run from
contracts/ directory (PowerShell environment)

Target
Current EverestVault / EverestVaultMulti codebase (Solc 0.8.24)

Command (conceptual)
slither .

Result

Slither completed its analysis without reporting issues for the configured detectors on this codebase (0 issues reported in the last run).

Interpretation
- Static analysis did not flag common categories of vulnerabilities on the analyzed contracts (e.g. reentrancy patterns, arithmetic issues under the current compiler settings, simple access-control mistakes).
- This is an automated layer of defense on top of unit tests and fuzzing. It is not a full manual audit.

4. Foundry Fuzz Test Suite Summary

Command
forge test -vv

Result
4 fuzz tests passing (256 runs per test, Solc 0.8.24)

Contract under fuzz
EverestVaultMulti (ETH-only vault model)

Tests
- testFuzz_DebtNeverNegative(uint96,uint96,uint96)
  Ensures user debt never becomes negative, even under randomized deposit / borrow / repay sequences.

- testFuzz_DepositBorrowRepay_ConservesEth(uint96,uint96)
  Checks ETH conservation over deposit + borrow + repay flows, accounting for protocol fees and internal accounting.

- testFuzz_DepositWithdraw_ConservesEth(uint96)
  Checks ETH conservation over deposit + withdraw flows, with the protocol fee applied on both deposit and withdrawal.

- testFuzz_LTVNeverAbove70(uint96,uint96)
  Ensures the loan-to-value ratio (debt / collateral) never exceeds 70% for any fuzzed sequence of deposits and borrows.

5. Manual & Functional Testing

In addition to automated tooling, the EverestVault code and deployment have undergone manual review and functional testing by the protocol authors.

Manual contract review
- Line-by-line review of EverestVault.sol focusing on:
  - Access control and owner / guardian powers.
  - Fee calculation paths on deposit and withdrawal.

  - Anti-loop same-block guard and lastBorrowBlock handling.
  - Pause behaviour, ensuring repay remains available while paused.
  - ETH handling, including revert paths and prevention of stuck funds.
- Review of interactions with OpenZeppelin libraries and their usage patterns (ownership, pausable logic, math).

Functional testing on testnet (Base Sepolia)
Using the GranEverest borrow app and direct calls:
- Deposits of ETH into the vault, verification of protocol fee accounting and user collateral balances.
- Borrow operations up to the enforced 70% LTV limit, including checks that LTV violations revert.
- Repay flows:
  - Exact repayment.
  - Overpayment with ETH refund.
- Withdrawals:
  - Full withdrawal with zero debt.
  - Partial withdrawal with existing debt, ensuring LTV remains within bounds.
- Pause / unpause:
  - Verification that deposit, borrow and withdraw revert while paused.
  - Verification that repay continues to work while the vault is paused.
- Cross-check of emitted events against expected values (collateral, debt, fees, and vault totals).

Mainnet smoke tests (Base)
On the production deployment with limited-size positions:
- Deposit, borrow, repay and withdraw flows executed end-to-end via the app and direct BaseScan interactions.
- Confirmation that:
  - Protocol fees are transferred to the configured feeRecipient.
  - User balances evolve as expected.
  - Revert conditions (e.g. attempting to exceed LTV) behave consistently with testnet and automated tests.

6. Conclusions

At the time of this report:
- All Hardhat unit tests pass (17/17).
- Slither static analysis completes with 0 reported issues on the current codebase.
- All Foundry fuzz tests pass (4/4, 256 runs each).
- Manual contract review and functional testing on Base Sepolia and Base mainnet confirm the expected behaviour of the ETH vault model in the tested scenarios.

Key properties validated
- Anti-loop protection
  Same-block borrow + deposit loops are blocked at the contract level.

- LTV limit
  Borrow operations are constrained to 70% of collateral; fuzzing confirms that the debt/collateral ratio does not exceed this limit under tested sequences.

- ETH conservation
  Deposit/withdraw and deposit/borrow/repay flows conserve ETH at the protocol level, apart from the configured protocol fee.

- Pause safety
  When paused, the vault blocks state-changing actions except repay, so users can still reduce risk and close positions while the protocol is paused.

Date: 2025-11-23