QORTRACE · SMART CONTRACT AUDIT

aave test

SECURITY	TRUST	FINDINGS
99	70	6
EXCELLENT	MODERATE	0C·0H·0M·1L·5I

TIER	STANDARD
LANGUAGE	SOLIDITY
SUBMISSION	GITHUB
MODEL	claude-sonnet-4-5-20250929
AUDIT ID	7d1abde1-3314-43f3-9621-4009de39492c

Executive summary

This is a test harness and interface suite for the Aave v3 protocol (Solidity 0.8.10). It includes verification contracts (e.g., ATokenHarness, PoolHarness, ReserveConfigurationHarness, VariableDebtTokenHarness, StableDebtTokenHarness), a simple ERC20 mock, a symbolic price oracle, and partial interface definitions. No production logic is implemented here—only wrappers and stubs for formal verification. The code is short (~60k chars) and primarily imports external Aave libraries. From a security standpoint, the harnesses themselves pose no direct risk if never deployed to production.

Findings by severity

Severity	Open	Resolved	Total
CRITICAL	0	0	0
HIGH	0	0	0
MEDIUM	0	0	0
LOW	1	0	1
INFO	5	0	5
TOTAL	**6**	**0**	**6**

Findings

LOW · 1

LOW SimpleERC20 transfer/transferFrom do not emit Transfer events

`certora/harness/SimpleERC20.sol:24–28`

SimpleERC20 is a minimal ERC20 stub without emit Transfer(from, to, amount). This violates the ERC20 standard and may cause off-chain tooling to misreport balances.

```
function transfer(address recipient, uint256 amount) external override returns (bool) {
    b[msg.sender] = sub(b[msg.sender], amount);
    b[recipient] = add(b[recipient], amount);
    return true;
}
```

FIX: Add emit Transfer(msg.sender, recipient, amount); and emit Approval(...) in approve() to comply with ERC20. Even for test harnesses, this ensures proper event logs.

INFO · 5

INFO Solidity 0.8.10 allows unchecked pragma version

`Multiple files (ATokenHarness.sol, PoolHarness.sol, etc.)`

The files use pragma solidity 0.8.10 (exact) or ^0.8.0 (floating). Exact pinning can prevent accidental usage of newer compiler versions with undiscovered bugs.

```
pragma solidity 0.8.10;
```

FIX: Pin to a single tested compiler version across all contracts (e.g., 0.8.10). If a floating pragma is used in interfaces, document that these are not production files.

INFO Public constructor in ATokenHarness without initialization checks

`certora/harness/ATokenHarness.sol:18`

ATokenHarness and other harness contracts declare public constructors that accept Pool instances but do not validate that the Pool address is non-zero.

```
constructor(Pool pool) public AToken(pool) {}
```

FIX: Add a require(address(pool) != address(0)) check in constructors if these harnesses might be instantiated outside of Certora test scenarios.

INFO ABIEncoderV2 pragma is redundant in Solidity >=0.8.0

`certora/harness/ReserveConfigurationHarness.sol:2`

ReserveConfigurationHarness declares pragma experimental ABIEncoderV2, which is the default in Solidity 0.8+. This line is unnecessary and can be removed.

```
pragma experimental ABIEncoderV2;
```

FIX: Remove the ABIEncoderV2 pragma for cleaner code.

INFO StableDebtTokenHarness overrides balanceOf to return static balance

`certora/harness/StableDebtTokenHarness.sol:16–18`

StableDebtTokenHarness's balanceOf skips interest accumulation (returns IncentivizedERC20.balanceOf(account)). This is a deliberate simplification for verification but deviates from production logic.

```
function balanceOf(address account) public view override returns (uint256) {
    return IncentivizedERC20.balanceOf(account);
}
```

FIX: Document clearly in comments that interest is intentionally zeroed for proof purposes. Do not deploy this harness to mainnet.

INFO ReserveConfigurationHarness exposes raw data field and setter helpers

`certora/harness/ReserveConfigurationHarness.sol:175–177`

ReserveConfigurationHarness exposes reservesConfig.data directly and provides setters (setLtv, setLiquidationThreshold, etc.). This is acceptable for testing but would allow unrestricted reconfiguration in a production context.

```
function getData() public view returns (uint256) {
    return reservesConfig.data;
}
```

FIX: Ensure this harness is never deployed in production. Add a comment that it is for formal verification only.

10-point trust check

Check	Result	Evidence
Access Control	PASS	Harness contracts do not implement privileged actions; ReservesSetupHelper uses Ownable for admin functions.
Events Emitted	FAIL	SimpleERC20 does not emit Transfer/Approval events, violating ERC20 standard.
Reentrancy Protection	PASS	No external calls in harness logic; underlying Aave libraries (not shown) are presumed protected.
Safe Math	PASS	Solidity 0.8.x has built-in overflow checks; SimpleERC20 add/sub helpers also revert on overflow.
Input Validation	FAIL	Harness constructors (e.g., ATokenHarness) do not validate pool != address(0).
No Hardcoded Secrets	PASS	No private keys, passwords, or API secrets in code.
No Tx Origin For Auth	PASS	No usage of tx.origin; msg.sender is used throughout.
Error Handling	PASS	Require statements present in SimpleERC20 add/sub; library calls are expected to revert on error.
Documentation Present	PASS	Harness contracts have brief NatSpec comments; interfaces are documented in Aave standard style.
Tests Referenced	FAIL	No unit tests or test scripts are included in this source code snippet.