ENIGMA DARK

Securing the Shadows



Invariant Testing Engagement

Twyne

Aave V3 AToken Wrapper

November, 2025

Contents

1. Summary
2. Engagement Overview
3. Risk Classification
4. Vulnerability Summary
5. Invariants and Postconditions Summary
6. Findings
7. Annex: Rounding Tolerance Analysis
8. Disclaimer

Summary

Enigma Dark
Enigma Dark is a web3 security firm leveraging the best talent in the space to secure all kinds of blockchain protocols and decentralized apps. Our team comprises experts who have honed their skills at some of the best auditing companies in the industry. With a proven track record as highly skilled white-hats, they bring a wealth of experience and a deep understanding of the technology and the ecosystem.

Learn more about us at enigmadark.com

Twyne: Aave V3 aToken Wrapper
The Aave V3 aToken Wrapper is a component of the Twyne protocol that provides an ERC4626-compliant vault interface for Aave V3 aTokens. It enables seamless integration of Aave's yield-bearing tokens within the Twyne credit delegation ecosystem.

This engagement utilizes a simplified version of the Enigma Dark invariant testing framework, focused on ERC4626 compliance and aToken-specific properties.

Engagement Overview

Over the course of 1 week, beginning November 13 2025, the Enigma Dark team conducted a continuous invariant testing engagement of the Aave V3 aToken Wrapper.

The engagement was performed by one Lead Security Researcher: vnmrtz.

The following repositories were reviewed at the specified commits:

Repository	Commit
0xTwyne/aave-v3-aToken-wrapper	`ae1ca5641f7c148f97084640017cf6140a899183`

Scope

```
src/
├── AaveV3ATokenWrapper.sol
└── CustomERC4626StataTokenUpgradeable.sol
```

Risk Classification

Severity	Description
Critical	Vulnerabilities that lead to a loss of significant portion of funds of the system.
High	Exploitable, causing loss or manipulation of assets or data.
Medium	Risk of future exploits that may or may not impact the smart contract execution.
Low	Minor code errors that may or may not impact the smart contract execution.
Informational	Non-critical observations or suggestions for improving code quality, readability, or best practices.

Vulnerability Summary

Severity	Count	Fixed	Acknowledged
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
Informational	0	0	0

Invariants and Postconditions Summary

This table provides an overview of all invariants and postconditions defined in the suite specification files.

Property Types

On this invariant testing framework there exists two types of Properties:

INVARIANTS (INV)

- These are properties that should always hold true in the system
- They are implemented under `/invariants` folder

POSTCONDITIONS

- These are properties that should hold true after an action is executed
- They are implemented under `/hooks` and `/handlers` folders
- There are two types of POSTCONDITIONS:

GLOBAL POSTCONDITIONS (GPOST)

- These are properties that should always hold true after any action is executed
- They are checked in the `_checkPostConditions` function within the HookAggregator contract

HANDLER-SPECIFIC POSTCONDITIONS (HSPOST)

- These are properties that should hold true after a specific action is executed in a specific context
- They are implemented within each handler function, under the HANDLER-SPECIFIC POSTCONDITIONS section

Table Structure

- **Property ID**: The unique identifier for each property
- **Description**: The detailed explanation of what the property should ensure
- **Status**: Current testing status (PASS/FAIL/PENDING)
- **Issues**: Related issues or notes when status is not PASS

Property ID	Description	Status	Issues
INV_ERC4626_A	The total value locked in the system in underlying assets should always equal or exceed the value redeemable by all share holders combined	PASS	-
INV_ERC4626_B	The sum of all individual share balances should equal the total supply	PASS	-
INV_ERC4626_C	totalSupply == 0 => totalAssets == 0	PASS	-
INV_ATOKEN_A	aToken.balanceOf(wrapper) + aToken.balanceOf(collateral vault) + 1 wei tolerance >= totalAssets()	PASS	-
INV_ATOKEN_B	aToken.scaledBalanceOf(wrapper) + aToken.scaledBalanceOf(collateral vault) >= totalSupply()	PASS	-
INV_AVAILABILITY_A	aaveV3ATokenWrapper.totalAssets() should never revert	PASS	-
INV_AVAILABILITY_B	aaveV3ATokenWrapper.latestAnswer() should never revert	PASS	-

Property ID	Description	Status	Issues
ERC4626_DEPOSIT_INVARIANT_A	maxDeposit MUST NOT revert	PASS	-
ERC4626_DEPOSIT_INVARIANT_B	previewDeposit MUST return close to and no more than shares minted at deposit if called in the same transaction	PASS	-
ERC4626_MINT_INVARIANT_A	maxMint MUST NOT revert	PASS	-
ERC4626_MINT_INVARIANT_B	previewMint MUST return close to and no fewer than assets deposited at mint if called in the same transaction	PASS	-
ERC4626_WITHDRAW_INVARIANT_A	maxWithdraw MUST NOT revert	PASS	-
ERC4626_WITHDRAW_INVARIANT_B	previewWithdraw MUST return close to and no fewer than shares burned at withdraw if called in the same transaction	PASS	-
ERC4626_REDEEM_INVARIANT_A	maxRedeem MUST NOT revert	PASS	-
ERC4626_REDEEM_INVARIANT_B	previewRedeem MUST return close to and no more than assets redeemed at redeem if called in the same transaction	PASS	-

Property ID	Description	Status	Issues
ERC4626_ROUNDTRIP_INVARIANT_A	redeem(deposit(a)) <= a	PASS	-
ERC4626_ROUNDTRIP_INVARIANT_B	s = deposit(a) s' = withdraw(a) s' >= s	PASS	-
ERC4626_ROUNDTRIP_INVARIANT_C	deposit(redeem(s)) <= s	PASS	-
ERC4626_ROUNDTRIP_INVARIANT_D	a = redeem(s) a' = mint(s) a' >= a	PASS	-
ERC4626_ROUNDTRIP_INVARIANT_E	withdraw(mint(s)) >= s	PASS	-
ERC4626_ROUNDTRIP_INVARIANT_F	a = mint(s) a' = redeem(s) a' <= a	PASS	-
ERC4626_ROUNDTRIP_INVARIANT_G	mint(withdraw(a)) >= a	PASS	-
ERC4626_ROUNDTRIP_INVARIANT_H	s = withdraw(a) s' = deposit(a) s' <= s	PASS	-

Property ID	Description	Status	Issues
GPOST_ERC4626_A	The exchange rate should increase monotonically	PASS	-
HSPOST_ATOKEN_A	After rebalance, the collateral vault's aToken scaled balance should equal the target shares	PASS	-
HSPOST_ATOKEN_B	During rebalance, totalSupply before == totalSupply after	PASS	-

Summary

- **Total Properties**: 26
- **Invariants (INV)**: 23
- **Global Postconditions (GPOST)**: 1
- **Handler-Specific Postconditions (HSPOST)**: 2
- **Status Distribution**:
    - PENDING: 0
    - PASS: 26
    - FAIL: 0

Categories

ERC4626 Compliance

- Total value locked vs redeemable value consistency
- Share balance accounting
- Empty vault state consistency
- Deposit/mint/withdraw/redeem preview functions
- Roundtrip operation guarantees

aToken Integration

- aToken balance tracking across wrapper and collateral vault
- Scaled balance consistency with total supply

Availability

- View functions must never revert

Exchange Rate

- Monotonic increase of assets/shares ratio

Rebalancing

- Collateral vault balance targeting
- Total supply preservation during rebalance

Status Definitions

- **PENDING**: Property has not been tested yet or testing is in progress
- **PASS**: Property passes all tests and behaves as expected

- **FAIL**: Property fails tests and requires attention

Resolution Options (for FAIL status properties)

- **FIXED**: The underlying issue has been resolved by the protocol team, property should pass on re-testing
- **ACKNOWLEDGED**: The issue has been reviewed and acknowledged by the protocol team as acceptable risk/design decision

Testing Workflow

1. Properties start as **PENDING** before testing
2. After initial testing, properties become **PASS** or **FAIL**
3. For **FAIL** status properties, the protocol team responds with either:
    - **FIXED**: Issue gets resolved (property should pass on re-testing)
    - **ACKNOWLEDGED**: Issue is accepted as acceptable risk/design decision

Findings

No vulnerabilities were identified during this engagement.

Annex: Rounding Tolerance Analysis

During initial invariant testing, several properties failed due to dust-level (1 wei) discrepancies caused by Aave's internal rounding mechanisms. After thorough analysis, these were determined to be expected behaviors inherent to how Aave's aToken math works, not security vulnerabilities. The invariants were adjusted with safe tolerances to account for these rounding characteristics.

Root Causes Identified

1. aToken Transfer Rounding (`depositATokens` path)

When users deposit aTokens directly via `depositATokens()` or `depositWithPermit(..., false)`:

- The wrapper's `previewDeposit()` uses `mulDiv(RAY, rate, Floor)` (rounds DOWN)
- Aave's aToken `_transfer` uses `rayDivCeil` (rounds UP)

This asymmetry causes the wrapper to receive 1 wei more scaled balance than shares minted. However, this is **protocol-favorable**, the extra dust accumulates in the wrapper, and users cannot extract it.

Testing confirmed: The `depositATokens` → `redeemATokens` cycle results in 1 wei of dust accumulating in the wrapper (protocol-favorable rounding). Users cannot extract value through this path.

2. Balance Splitting via `rayMulFloor`

When aToken balances are split between the wrapper and collateral vault (via `rebalanceATokens_CV`):

- Aave's `balanceOf()` uses `rayMulFloor` (rounds DOWN) on each address separately
- `floor(a) + floor(b)` can be up to 1 wei less than `floor(a + b)`

This means the sum of individual `balanceOf` calls can be 1 wei less than if all tokens were in a single address, a mathematical property of floor division, not a vulnerability.

Invariant Adjustments

Property	Change	Reason
INV_ERC4626_A	Added `+ 1` wei tolerance	rayMulFloor splitting
INV_ATOKEN_A	Added `+ 1` wei tolerance	rayMulFloor splitting
INV_ATOKEN_B	Changed `==` to `>=`	depositATokens extra scaled balance

Security Implications

No exploit vectors were identified during this testing engagement. All rounding behaviors observed are:

1. **Protocol-favorable**: Dust accumulates in the wrapper, not extractable by users
2. **Bounded**: Maximum discrepancy is 1 wei per operation
3. **Consistent with Aave**: Same behavior exists in Aave's original StataToken implementation

Recommendation

The 1 wei tolerances are safe and appropriate for production. Integrators should be aware that:

- Direct aToken deposits (`depositATokens`) may deposit 1 wei more scaled balance than shares
  minted.
- Standard `deposit()` path (which supplies to Aave first) has exact accounting
- No value extraction paths were identified through invariant testing

Disclaimer

This report does not endorse or critique any specific project or team. It does not assess the economic value or viability of any product or asset developed by parties engaging Enigma Dark for security assessments. We do not provide warranties regarding the bug-free nature of analyzed technology or make judgments on its business model, proprietors, or legal compliance.

This report is not intended for investment decisions or project participation guidance. Enigma Dark aims to improve code quality and mitigate risks associated with blockchain technology and cryptographic tokens through rigorous assessments.

Blockchain technology and cryptographic assets inherently involve significant risks. Each entity is responsible for conducting their own due diligence and maintaining security measures. Our assessments aim to reduce vulnerabilities but do not guarantee the security or functionality of the technologies analyzed.

This security engagement does not guarantee against a hack. It is a review of the codebase during a specific period of time. Enigma Dark makes no warranties regarding the security of the code and does not warrant that the code is free from defects. By deploying or using the code, the project and users of the contracts agree to use the code at their own risk. Any modifications to the code will require a new security review.