yAudit

December 9, 2025

Prepared for
Twyne

Audited by
Adriro
HHK

Twyne AAVE operators

Smart Contract Security Assessment

Contents

1 Review Summary

1.1 Protocol Overview

Twyne is a credit delegation protocol that lets borrowers rent unused borrowing power from other lenders to boost their Liquidation LTV. Lenders earn additional yield while borrowers get to ramp up their leverage or insulate their debt.

1.2 Audit Scope

This audit covers 5 smart contracts totaling approximately 541 lines of code across 1 day of review.

```
0xTwyne/twyne-contracts/src
└── operators
    ├── LeverageOperator_EulerFL.sol
    ├── DeleverageOperator_EulerFL.sol
    ├── AaveV3LeverageOperator.sol
    ├── AaveV3DeleverageOperator.sol
    └── AaveV3TeleportOperator.sol
```

1.3 Risk Assessment Framework

1.3.1 Severity Classification

1.4 Key Findings

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	1
Informational	4

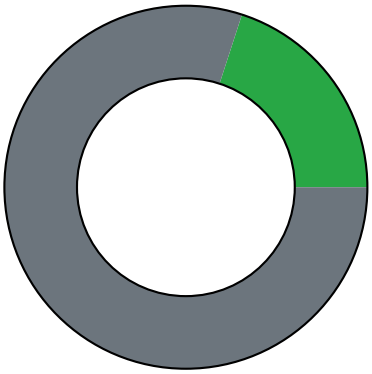

Figure 1: Distribution of security findings by impact level

1.5 Overall Assessment

The audit of Twyne's Aave operators identified no critical or high-severity vulnerabilities, indicating a solid implementation that maintains the security standards of the protocol's foundation. The informational and gas findings outlined mostly minor improvements, with the

Severity	Description	Potential Impact
Critical	Immediate threat to user funds or protocol integrity	Direct loss of funds, protocol compromise
High	Significant security risk requiring urgent attention	Potential fund loss, major functionality disruption
Medium	Important issue that should be addressed	Limited fund risk, functionality concerns
Low	Minor issue with minimal impact	Best practice violations, minor inefficiencies
Undetermined	Findings whose impact could not be fully assessed within the time constraints of the engagement. These issues may range from low to critical severity, and although their exact consequences remain uncertain, they present a sufficient potential risk to warrant attention and remediation.	Varies based on actual severity
Gas	Findings that can improve the gas efficiency of the contracts.	Increased transaction costs
Informational	Code quality and best practice recommendations	Reduced maintainability and readability

Table 1: severity classification

exception of one finding where the Euler controller was not deactivated properly. No funds were at risk from this issue.

2 Audit Overview

2.1 Project Information

Protocol Name: Twyne
Repository: https://github.com/0xTwyne/twyne-contracts
Commit Hash: c9164fbb0fda1d9a757e5f790d1aea92eda963b1
Commit URL: c9164fbb0fda1d9a757e5f790d1aea92eda963b1

2.2 Audit Team

Adriro, HHK

2.3 Audit Timeline

The audit was conducted from November 24 to 24, 2025.

2.4 Audit Resources

Code repositories and documentation

2.5 Critical Findings

None.

2.6 High Findings

None.

2.7 Medium Findings

None.

2.8 Low Findings

2.8.1 Incorrect disable controller call

Technical Details

The leverage and deleverage operators that use Euler as the flashloan provider first enable the controller via `EVC.enableController()` and then disable it via `EVC.disableController()`, which is incorrect, since the controller must disable itself.

```
120:         eulerFL_items[0] = IEVC.BatchItem({
121:             targetContract: address(EULER_EVC),
122:             onBehalfOfAccount: address(0),
123:             value: 0,
124:             data: abi.encodeCall(IEVC.enableController, (address(this), address(
targetVault)))
125:         });
...          ...
147:         // 5) Disable controller
148:         eulerFL_items[4] = IEVC.BatchItem({
149:             targetContract: address(EULER_EVC),
150:             onBehalfOfAccount: address(0),
151:             value: 0,
152:             data: abi.encodeCall(IEVC.disableController, (address(this)))
153:         });
```

Impact

Low. The controller is not properly disabled and will remain enabled after the leverage or deleverage call.

Recommendation

In `LeverageOperator_EulerFL` and `DeleverageOperator_EulerFL`, change the last element of the Euler batch and call `disableController()` directly through the `targetVault` / `collateralAsset`.

Developer Response

Fixed in PR#204.

2.9 Gas Savings Findings

2.9.1 Potential zero value transfers in `DeleverageOperator_EulerFL`

Technical Details

The implementation of `executeDeleverage()` returns the leftovers to the caller without checking if the actual amounts are zero.

```
134:         IERC20(targetAsset).safeTransfer(msgSender, IERC20(targetAsset).balanceOf(
address(this)));
135:         IERC20(underlyingCollateral).safeTransfer(msgSender, IERC20(
underlyingCollateral).balanceOf(address(this)));
```

Impact

Gas savings.

Recommendation

Check if balances are zero before transferring the assets.

Developer Response

Acknowledged.

2.10 Informational Findings

2.10.1 Unnecessary 10 wei addition to flashloan amount

Technical Details

The `executeTeleport()` function inside `AaveV3TeleportOperator` adds `10 wei` to the Morpho flashloan amount. It's unclear why this value is added, especially if a user already provides the full debt amount or `type(uint256).max`.
Inside `onMorphoFlashLoan()`, `debtAmount` is used for repayment, which ignores this 10 wei addition, resulting in useless extra wei being flashloaned.

Impact

Informational.

Recommendation

Remove the `+ 10` from the call.

Developer Response

Fixed in PR#205.

2.10.2 Morpho references in Euler flashloan operators

Technical Details

The `LeverageOperator_EulerFL` and `DeleverageOperator_EulerFL` contracts refer to Morpho when checking the caller in the flashloan callbacks.

- `LeverageOperator_EulerFL.sol#L169`
- `DeleverageOperator_EulerFL.sol#L151`

Impact

Informational.

Recommendation

Rename the error.

Developer Response

Fixed in PR#207.

2.10.3 Unbounded debt repayment in AaveV3TeleportOperator

Technical Details

The teleport operation takes the `debtAmount` as user input and forwards that amount when repaying the debt using `AAVE_POOL.repay()`. Given that the inner implementation of `repay()` silently limits the repayment amount to the actual debt, using a higher amount would result in a leftover in the operator.

Impact

Informational.

Recommendation

Cap `debtAmount` to the user's current debt.

Developer Response

Fixed in PR#208.

2.10.4 Inaccurate comment in `DeleverageOperator_EulerFL`

Technical Details

The following comment in `onFlashLoan()` hints that an automatic pull of assets would settle the Euler debt. However, repayment is explicitly performed using `collateralAsset.repay()`.

```
177:         // Collateral asset will automatically pull the repayment amount
```

Impact

Informational.

Recommendation

Remove the comment.

Developer Response

Fixed in PR#206.

2.11 Final Remarks

The review focused on new operators for AAVE as well as rework on Euler operators. The AAVE operators will be a welcome addition to Twyne's new AAVE integration, while the Euler operators update will allow the use of Euler directly for flashloans instead of Morpho as in the previous version. The review did not discover any medium-severity or above issues, highlighting the seriousness of the Twyne team.