yAudit

February 2026

Prepared for
Twyne

Audited by
Adriro
HHK

Twyne 2026-Q1 update

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 7 pull requests across 2.5 days of review.

```
0xTwyne/twyne-contracts/src
├── TwyneFactory
│   ├── BridgeHookTarget.sol
│   └── CollateralVaultFactory.sol
└── twyne
    ├── AaveV3CollateralVault.sol
    ├── CollateralVaultBase.sol
    ├── EulerCollateralVault.sol
    ├── IRMTwyneCurve.sol
    ├── IRMTwyneCurveGamma32.sol
    └── VaultManager.sol
```

1.3 Risk Assessment Framework

1.3.1 Severity Classification

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

1.4 Key Findings

Breakdown of Finding Impacts

Impact Level	Count
Critical	0
High	0
Medium	0
Low	1
Informational	5

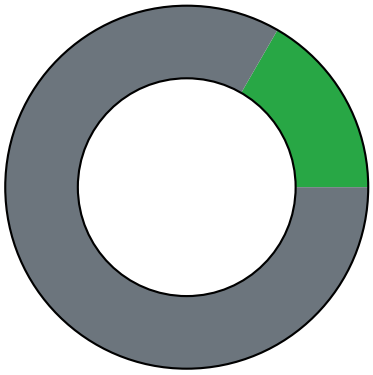

Figure 1: Distribution of security findings by impact level

1.5 Overall Assessment

The audit of Twyne's 2025 Q1 update identified no medium severity or above vulnerabilities, indicating a solid update that maintains the security standards of the protocol's foundation. The updates result in three major changes: a refactor of the IRM, usage of the intermediate vault address over collateral inside the buffer mapping, and a rework of the fixed Twyne LTV into a dynamic one.

2 Audit Overview

2.1 Project Information

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

- ff79392ec158c3ca46830120f26548056a55c202
- 3cc102243d24cdcda85b7cc2b9dfc65ab15f2d49
- 326cc7a018e2925e61640db4faf18a4bbf822256
- bbe95268ed23a2e2be059543486f92c55db42f1e
- 7cf2f5ad85f5d9c307b761c713bd10811a7de074
- 9aaeae6540ea87ee20784dda9bd82d0e54d643f8
- 1149c13c8e08786ed7f39c965915ebe95ae01976
- 7ba36ceafa19a0434cde1b8dc9fe719de033c148
- fcf76877efe06fad17be6890941b62a87cba5b71
- f15c596c527d9865d95c2fe90f06d0367435fba7

2.2 Audit Team

Adriro, HHK

2.3 Audit Timeline

The audit was conducted from February 6 to 9, 2026.

2.4 Audit Resources

- Code repositories and documentation
- Dynamic Liquidation LTV whitepaper

2.5 Critical Findings

None.

2.6 High Findings

None.

2.7 Medium Findings

None.

2.8 Low Findings

2.8.1 Reducing `maxTwyneLTV` can retroactively liquidate existing positions

When `maxTwyneLTVs` is lowered by governance or when the vault implementation is updated, positions that were healthy under the previous parameter may become immediately liquidatable.

Technical Details

In `_collateralScaledByLiqLTV1e8()`, the effective liquidation LTV is capped at `Math.min(twyneLiqLTV, twyneVaultManager.maxTwyneLTVs(address(intermediateVault)))`. This value is read from `twyneVaultManager` at query time, meaning any reduction to `maxTwyneLTVs` takes effect immediately for all existing positions.
A borrower who opened a position at an LTV just below the previous `maxTwyneLTV` would find their position suddenly above the new threshold, making them liquidatable. The same applies if the vault implementation is upgraded and the new logic changes how `twyneLiqLTV` interacts with `maxTwyneLTVs`.

Impact

Low. This requires a governance action to reduce `maxTwyneLTVs`, which is a privileged operation. However, the absence of a buffer or time-delay mechanism means the effect on existing borrowers is immediate and potentially severe.

Recommendation

Consider implementing a time-delayed parameter change or a grace period during which existing positions are evaluated against the old `maxTwyneLTV` while new positions use the updated value. Additionally, assess the effects of upgrading the implementation over existing collateral vaults.

Developer Response

Fixed in PR#239.
yAudit: Before deploying the ramp-down feature, verify no existing vault has twyneLiqLTV > maxTwyneLTVs on-chain, as those vaults would have already had their effective LTV reduced instantly by a prior maxTwyneLTVs change and the ramp mechanism cannot retroactively protect them.

2.9 Gas Savings Findings

None.

2.10 Informational Findings

2.10.1 No way to deregister an intermediate vault

Technical Details

`VaultManager.setIntermediateVault()` sets `isIntermediateVault[vault] = true` but there is no function to set it back to `false`. Once registered, an intermediate vault cannot be removed from the approved set.

Impact

Informational. If an intermediate vault needs to be deprecated, it will remain permanently valid in `isIntermediateVault`.

Recommendation

Add a `bool value` parameter to the function and set the mapping with it.

Developer Response

Fixed in PR#252.

2.10.2 `_handleExcessCredit` duplicates identical logic across Euler and Aave vaults

Technical Details

The credit handling logic in EulerCollateralVault.sol#L87 and AaveV3CollateralVault.sol#L140 is identical. The only difference is the Aave override mixing in `rebalanceATokens_CV`, which is aToken wrapper bookkeeping — not credit handling.

Impact

Informational. Duplicated logic increases divergence risk — PR #243's bugfix had to be applied to both files independently.

Recommendation

Move the shared logic into `CollateralVaultBase`. Aave can overrides with a simple `super._handleExcessCredit(...)` + `rebalanceATokens_CV`.

Developer Response

Acknowledged.

2.10.3 Stale NatSpec on `_invariantCollateralAmount()` does not reflect dynamic liquidation LTV

The `@dev` comment on `_invariantCollateralAmount()` describes the formula as `ceil(userCollateral * twyneLiqLTV * MAXFACTOR / (externalLiqLTV * buffer))`, but the actual computation now uses a dynamic liquidation LTV model introduced with the Dynamic Liquidation LTV update.

Technical Details

The NatSpec comment on `_invariantCollateralAmount()` in `CollateralVaultBase` references an older, static formula. The current implementation uses a dynamic LTV derived from the credit reservation invariant (`C · λ^dyn_t = β_safe · λ̃e · (C_LP + C)`), as implemented in `_collateralScaledByLiqLTV1e8()`.

Impact

Informational.

Recommendation

Update the `@dev` comment on `_invariantCollateralAmount()` to reflect the dynamic liquidation LTV formula currently in use.

Developer Response

Fixed in PR#242.

2.10.4 Oracle quotes scaled collateral in EulerCollateralVault

The `_collateralScaledByLiqLTV1e8()` return value is passed directly into `EulerRouter.getQuote()` as the input amount. Since this value is already scaled by the LLTV (in 1e8 precision), the oracle must price it linearly — any non-linear pricing behavior would produce incorrect collateral valuations.

Technical Details

In `EulerCollateralVault._canLiquidate()`, the collateral value used for liquidation checks is computed by calling `getQuote()` with the output of `_collateralScaledByLiqLTV1e8(false, adjExtLiqLTV)`. This value equals the user's collateral multiplied by a liquidation LTV factor at 1e8 precision. The oracle receives this scaled quantity and is expected to convert it to the unit of account.
If the oracle's `getQuote()` implementation is not linear in the input amount (i.e., `getQuote(k * x) != k * getQuote(x)`), the resulting collateral valuation will be incorrect.

Impact

Informational.

Recommendation

Document the linearity requirement for Oracles used with `getQuote()` in this context. Consider adding a comment or NatSpec annotation at the oracle configuration point to alert of this behavior.

Developer Response

Fixed in PR#242.

2.10.5 IRMTwyneCurve constructor reverts with misleading error

The `IRMTwyneCurve` constructor reuses the `E_IRMUpdateUnauthorized()` error from the Euler Vault Kit's `IIRM` interface to signal invalid constructor parameters, which is semantically incorrect and can mislead developers and monitoring tools during debugging.

Technical Details

The constructor validates that `polynomialParameter_` and `nonlinearPoint_` are non-zero and that `nonlinearPoint_` is below 100%. When any of these checks fail, it reverts with `E_IRMUpdateUnauthorized()`:

```
if (
    polynomialParameter_ == 0
    || nonlinearPoint_ == 0
    || nonlinearPoint_ >= 1e18
) revert E_IRMUpdateUnauthorized();
```

This error is defined in the `IIRM` interface and is intended for authorization failures, specifically when `msg.sender` is not the expected vault (as seen in `computeInterestRate()`). Using it for parameter validation conflates two distinct failure modes: unauthorized access and invalid input. Off-chain tooling or integrators catching `E_IRMUpdateUnauthorized` would incorrectly interpret a deployment misconfiguration as an authorization issue.

Impact

Informational.

Recommendation

Define a dedicated error (e.g., `error E_InvalidIRMParameters()`) for the constructor validation and revert with that instead.

Developer Response

Acknowledged.

2.11 Final Remarks

Twyne's 2025 Q1 update introduces changes to the IRM, LTV mapping, and liquidation mechanism, no medium or above severity findings were found outlying solid development practices from the Twyne team. The dynamic liquidation LTV is a welcome change that will result in a better user experience. Some concerns were raised about the LTV potentially being manipulated using the cash balance on the intermediate vault, but this did not result in any meaningful threat and is a necessary trade-off for the feature. All issues identified during the review have been acknowledged or fixed. The final commit containing these changes on the private Twyne repository can be found at f15c596c527d9865d95c2fe90f06d0367435fba7.

Twyne 2026-Q1 update

Completed 2026-02-09