Contents

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 v1
Twyne is a credit delegation protocol that turns unused borrowing power into yield. It maximizes lending efficiency by delegating unused borrowing power to users that want to increase leverage or protect against liquidation.

Engagement Overview

Over the course of 3 weeks, beginning March 31 2025, the Enigma Dark team conducted a continous invariant testing engagement of the Twyne v1 project. The engagement was performed by one Lead Security Researcher: vnmrtz.

The following repositories were reviewed at the specified commits:

Repository	Commit
0xTwyne/twyne-contracts-v1	`deca583d97e11f3c0ad2140401d01a9b3f863a41`

Risk Classification

Severity	Description
Critical	Vulnerabilities that lead to a loss of a 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	1	1	0
Low	2	0	2
Informational	1	1	0

Findings

Index	Issue Title	Status
M-01	Denial of service in liquidations if collateral vault remains open long enough	Fixed
L-01	Max repayment may fail after decrease in collateral asset price	Acknowledged
L-02	Increasing external liquidation buffer can violate collateral vault's minimum LTV	Acknowledged
I-01	Minor improvements to code and comments	Fixed

Detailed Findings

High Risk

No issues found.

Medium Risk

M-01 - Denial of service in liquidations if collateral vault remains open long enough

Severity: Medium Risk

Context:

- EulerCollateralVault.sol#L116
- EulerCollateralVault.sol#L143

Technical Details:
In lending protocols, it is critical to maintain the availability property that ensures liquidations are always available for underwater positions in the case of Twyne invariant suite this is implemented with the invariant `INV_CV_NR_D: canLiquidate must never revert`.

However, in Twyne's implementation, if a collateral vault debt position remains open long enough, the debt owed to the intermediate vault can eventually grow larger than `totalAssetsDepositedOrReserved`. When this happens, the calculation `totalAssetsDepositedOrReserved - maxRelease()` underflows and causes the transaction to revert.

Impact:
If the collateral vault is left unchecked over time, liquidations can become unavailable, potentially leading to the accumulation of bad debt in the system.

Recommendation:
Consider updating `maxRelease` to return the minimum of the intermediate vault's debt and `totalAssetsDepositedOrReserved` to prevent the underflow condition.

Developer Response:
Fixed at commit `4888034`.

`maxRelease` now correctly returns the minimum value between `intermediateVault.debtOf(address(this))` `totalAssetsDepositedOrReserved`, ensuring that the `INV_CV_NR_D` invariant consistently holds.

Low Risk

L-01 - Max repayment may fail after decrease in collateral asset price

Severity: Low Risk

Context:

- EulerCollateralVault.sol#L65-L69
- CollateralVaultBase.sol#L176-L187

Technical Details:
In lending protocols, it is critical to maintain the availability property that ensures users and integrators can always repay the maximum allowable amount without failure. Specifically, the invariant `HSPOST_CV_NR_D: repay(maxRepay) must never revert` must consistently hold.

However, in the case of Twyne, this invariant breaks when the value of the collateral drops below a certain threshold.

PoC:

```
    // HSPOST_CV_NR_D

    function assert_HSPOST_CV_NR_D() external
setupActor(collateralVaultUser) {
        bool success;
        bytes memory data;

        uint256 maxRepay = userCollateralVault.maxRepay();
        require(maxRepay != 0, "maxRepay should not be 0");

        uint256 collateralBalanceOfBorrower =
assetTST.balanceOf(collateralVaultUser);

        // Ensure borrower has enough assets for repayment
        if (collateralBalanceOfBorrower < maxRepay) {
            assetTST.mint(collateralVaultUser, maxRepay -
collateralBalanceOfBorrower);
        }

        target = address(userCollateralVault);

        _before();
        (success, data) = actor.proxy(target,
abi.encodeCall(CollateralVaultBase.repay, maxRepay));

        assertTrue(success, HSPOST_CV_NR_D);

        _after();
    }
```

```
    // Replay test

    function test_replay_assert_HSPOST_CV_NR_D() public {
        Tester.mint(2739, 6, 1);
        Tester.setPrice(100000, 0);
        Tester.mint(1, 0, 0);
        Tester.deposit(1174, 0, 2);
        Tester.depositCV(12);

Tester.borrowCV(115792089237316195423570985008687907853269984665640564039457
584007913129639935, 0);
        Tester.setPrice(1, 1);
        Tester.assert_HSPOST_CV_NR_D();
    }
```

Impact:
Users and integrators are unable to repay the maxRepay amount if the value of collateral is too low.

Recommendation:
Add additional logic to handle the edge case of large price drops in collateral assets to avoid this revert case.

Developer Response:
Acknowledged. Our plan is to check all the operations on twyne when collateral asset price drops to a very low value. We'll compare the execution against our expectation. Our immediate reponse to repay() failing is: if collateral is worthless, no rational borrower will repay the debt. We can change the invariant for now to avoid this case.

Twyne will only support blue chip assets as collateral for the short term, which makes this a highly unlikely scenario.

L-02 - Increasing external liquidation buffer can violate collateral vault's minimum LTV

Severity: Low Risk

Context:

- VaultManager.sol#L98-L101

Technical Details:
The `VaultManager` contract allows governance to update the `externalLiqBuffer` for a collateral asset via the `setExternalLiqBuffer` function. However, in certain scenarios, increasing this buffer can cause the collateral vault's Loan-to-Value (LTV) ratio to fall below the allowed minimum, violating system constraints and making the vault liquidatable.

PoC:

```
    // INV_CV_BASE_A

    function assert_INV_CV_BASE_A() internal {
        uint256 liqLTV = userCollateralVault.twyneLiqLTV();
        uint256 minLTV = eTST.LTVLiquidation(address(eTST2));
        uint256 externalLiqBuffer =
vaultManager.externalLiqBuffers(address(eTST2));
        uint256 maxTwyneLiqLTV = vaultManager.maxTwyneLTVs(address(eTST2));

        assertGe(liqLTV * 1e4, minLTV * externalLiqBuffer, INV_CV_BASE_A);
        assertLe(liqLTV, maxTwyneLiqLTV, INV_CV_BASE_A);
    }
```

```
    // Replay test

    function test_replay_setExternalLiqBuffer() public {
        _setUpActor(USER1);
        Tester.setExternalLiqBuffer(0);
        Tester.setTwyneLiqLTV(8585);
        Tester.setExternalLiqBuffer(40);
        assert_INV_CV_BASE_A();
    }
```


Impact:
Collateral vault's LTV can go below its allowed minimum.

Recommendation:
Introduce an additional validation in the `setExternalLiqBuffer` function to ensure that increasing the `externalLiqBuffer` does not reduce the vault's LTV below the allowed minimum. The function should revert if this condition is not met.

Developer Response:
Acknowledged. It may be better to leave this test as is and and expect a revert. During the whitelist period, we'll discuss with the team if some mechanism to handle this change in external liquidation buffer is needed.

Most lending protocols have a similar situation where governance can change the allowed LTV values to a point which can cause some users to be liquidated

Informational

I-01 - Minor improvements to code and comments

Severity: Informational

Context: See below.

Technical Details:

1. EulerCollateralVault.sol#L86 - The NatSpec comment incorrectly uses the word “help” instead of “held” when referring to the collateral assets.
2. CollateralVaultBase.sol#L385-L388 - Consider caching the result of `_invariantCollateralAmount` in a local variable to avoid redundant external calls.

Developer Response:
Fixed at commit `ff27a9e`.

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.