ENIGMA DARK

Securing the Shadows

Invariant Testing Engagement

Euler Labs (EVK)

March, 2024

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 decentralised 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

Euler Labs
Euler Labs is a team of developers and quantitative analysts building DeFi applications for the future of finance. Which have developed Euler v2, a modular lending protocol that uses the Euler Vault Kit (EVK).

Codebase: Euler vault kit
Euler v2 is a system of ERC-4626 vaults built using a custom-built vault development kit, called the EVK, and chained together using the EVC. The Euler Vault Kit (EVK), is a set of tools which empowers builders to deploy and chain together their own customised lending vaults in a permissionless manner.

Engagement Overview: Continuous Invariant Testing

Over the course of 6 weeks, Enigma Dark's lead security researcher, Victor Martinez, conducted a continuous invariant testing engagement on Euler's EVK. The engagement consisted in three phases:

- Build an exhaustive fuzzing suite tailored the Euler Vault Kit.
- Identify and check +55 invariant properties for the system assesing issues found.
- Iterate through the development cycle to ensure the invariants are maintained after audit & development changes.

During this period +55 invariants were checked against every iteration of the system using top-notch tooling and millions of runs. A total of 3 impactful issues were found and reported to the Euler team, at the time of this report all properties have been checked.

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.

Vulnerability Summary

Severity	Count	Fixed	Acknowledged
Critical	0	0	0
High	2	2	0
Medium	1	1	0
Low	0	0	0

Findings

High Findings

[H-01] - Value can be extracted from the vault

Severity: High Risk

Technical Details: The current calculation method for determining liability value of an account in the protocol involves a division by a unit scale of 1e18, which results in values smaller than 1e18 being rounded to 0. This allows non-collateralized accounts to extract value from the protocol borrowing small amounts if the resulting liability value of the account is less than 1e18. This leads to a violation of the invariant BM_INVARIANT_O: `debt(user) != 0 => vault.balanceOf(user) != 0`.

Impact: This issue enables the extraction of value from the protocol in small increments, which could potentially be exploited for larger attacks.

Proof of Concept: The test `CryticToFoundry::test_BM_INVARIANT_O_ROUNDING` demonstrates how 1 wei can be borrowed from the protocol at a time without requiring collateral.

Recommendation: Two potential solutions are proposed:

1. **Rounding Up**: Consider rounding up the liability value calculation to prevent values smaller than 1e18 from being rounded to 0.

2. **Collateral Check**: Implement a check to disallow accounts without collateral from withdrawing funds from the protocol. This would ensure that only collateralized accounts can borrow from the protocol.

Resolution: Fixed.

[H-02] - DOS on repayments after reducing supply cap below current supply

Severity: High Risk

Technical Details: `RiskManager::107` implements a check that makes sure after reducing supply caps below the current supply, this one can only stay equal or decrease. This should not be a problem, since it shouldn't affect any borrowing operations within the vault.

```
if (supply > vaultCache.supplyCap && supply > prevSupply) revert
E_SupplyCapExceeded();
```

However trying to repay while current supply is over the caps causes the execution to revert due to the aforementioned check.

The test `test_assert_BM_INVARIANT_P` shows one example of this behaviour when a user tries to repay an asset:

BEFORE REPAY:

```
prevSupply = 4477916
prevCash = 0
prevBorrows = 4477916
```

AFTER REPAY:

```
supply = 4477917
cash = 301
borrows = 4477616
```

We can see that even though no operation to increase supply has been called, `supply` has increased by 1 wei (making the check and execution revert). This is due to the rounding applied to owed amounts in `decreaseBorrow` function that acts as rounding down the amount of debt repaid therefore increasing totalAssets (`supply` in the context of the vault checks).

```
Assets owed = owedExact.toAssetsUp();
```

This issue violated a core invariant of lending protocols `BM_INVARIANT_P: a user can always repay debt in full` , as well as the liveness property of repayments. Leaving the system at higher risk of entering a bad debt cycle.

Impact: High, repayments enter DOS state After reducing supply cap below current supply, leaving users unable to repay their debt. Extremely dangerous on periods of high utilization.

Proof of Concept: The test `CryticToFoundry::test_assert_BM_INVARIANT_P` shows the behaviour of the system on this scenario.

Recommendation: A potential solution is avoid checking for supply increases on repayments/borrows. Similar to other lending protocols.

Resolution: Fixed.

Medium Findings

[M-01] - Share value is diluted with each fee conversion

Severity: Medium Risk

Technical Details: During the execution of the `convertFees` function, according to the protocol's intended behavior, `totalShares` should be decreased by the accumulated fee amount to be later increased during the minting of Etokens to `protocolFeeReceiver` and `feeReceiver`. However, due to a flaw in the implementation, the cached value of accumulated fees is being set to 0 in the step above. As a result, new shares are being minted without proper deduction, diluting users' positions unexpectedly. This issue leads to a violation of the invariant TM_INVARIANT_C: `totalSupply == sum of balanceOf(actors) + accumulatedFees`.

Impact: This issue has a significant impact on users' positions within the protocol. The unexpected minting of new shares without corresponding deductions dilutes the positions of existing users, potentially leading to loss of value.

Proof of Concept: The test `CryticToFoundry::test_2TM_INVARIANT_C` demonstrates how after a fee conversion totalShares is not properly updated leading to a violation of the invariant.

Recommendation: Change the order of the operations in the `convertFees` function to ensure that the accumulated fees are properly deducted from `totalShares` before setting the cached value to 0.

Resolution: Fixed.

Disclaimer

This report is based on continuous invariant testing over a 6-week period, focusing on improving the integrity and security of Euler's EVK.

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 at a 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, Euler 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.