



# Contents

1. Summary
2. Engagement Overview
3. Risk Classification
4. Vulnerability Summary
5. Findings
6. 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](https://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 Earn**

The Euler Earn project is an open source protocol for permissionless risk curation on top of ERC4626 vaults(strategies). Although it is initially designed to be integrated with Euler V2 vaults, technically it supports any other vault as long as it is ERC4626 compliant.

# Engagement Overview

Over the course of 2.6 weeks starting October 1st 2024, Enigma Dark's lead security researcher, Victor Martinez, conducted a continuous invariant testing engagement on Euler Earn. The engagement consisted in three phases:

- Build an exhaustive fuzzing suite tailored the Euler Earn codebase.
- Identify and check +40 invariant properties for the system assessing issues found.
- Iterate through the development cycle to ensure the invariants are maintained after audit & development changes.

During this period +40 invariants were checked against every iteration of the codebase using top-notch tooling and millions of runs. A total of 2 impactful issues and 2 minor issues were found and reported to the Euler team, at the time of this report all properties have been checked against the specified commit:

Repository	Commit
euler-xyz/euler-earn	617f3aa4ca138c026a87a002f0da98b24b807f2a

## 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	1	1	0
Medium	1	1	0
Low	1	0	1
Informational	1	1	0

## Findings

Index	Issue Title	Status
H-01	Earn Vault can be bricked via strategy share donation before rebalancing	Fixed
M-01	Missing check <code>feeShares != 0</code> in <code>_previousHarvestBeforeWithdraw()</code>	Fixed
L-01	ERC4626 compliancy	Acknowledged
I-01	Earn Vault does not execute gulp after enabling back an strategy from emergency to active	Fixed

# Detailed Findings

## High Risk

### H-01 - Earn Vault can be bricked via strategy share donation before rebalancing

**Severity:** High Risk

**Technical Details:**

The Earn Vault is vulnerable to being "bricked" if a strategy share donation occurs before the vault rebalances into the donated strategy. This results in a violation of the following invariant:

```
ERC4626_REDEEM_INVARIANT_C: maxRedeem MUST return the maximum amount of shares that could be transferred from owner through redeem without causing a revert.
```

The issue arises from the following sequence of steps, starting with the initial setup: the vault is configured with two strategies, eTST at index 0 and eTST2 at index 1 in the withdrawal queue.

1. User deposit into Earn.
2. Rebalance into the second strategy only.
  - Rebalance into the second strategy only.
  - 2500 total points; 1000 for reserve(40%), 500(20%) for eTST, 1000(40%) for eTST2.
  - 10k deposited; 6000 for reserve, 4000 for eTSTsecondary as we are not rebalancing into eTST .
3. Someone donates shares of eTST to Earn vault
  - At this point, the Earn vault is unaware of the donated 1 share of eTST since it hasn't been rebalanced into the strategy yet.
  - When a harvest is triggered during the redeem process, the 1 donated share won't be accounted for because strategies with a 0 allocated amount are ignored during harvest.
  - However, the strategy is not ignored during the withdrawal process in `_withdraw()`. As a result, the following operation: `$. strategies[ address(strategy) ]. allocated -= uint120(withdrawAmount)` will trigger a revert due to an underflow, as it attempts to subtract 1 from 0.

**Impact:**

High, the Euler Earn Vault can be bricked by an external donation of strategy shares before rebalancing into that strategy. This renders the vault unusable, as the redeem flow will fail due to incorrect handling of unallocated shares.

**Recommendation:**

Remove the `strategyAllAllocatedAmount == 0` check in the harvesting logic, allowing harvesting from strategies even with zero allocated amounts.

**Developer Response:**

Fixed at [PR #129](#).

Now, harvesting a strategy even with 0 allocated amounts is allowed, enabling the collection of donated strategy shares when a donation occurs before rebalancing into that strategy. The harvested shares will be accounted for as positive yield.

## Medium Risk

### M-01 - Missing check `feeShares != 0` in

`_previewHarvestBeforeWithdraw()`

**Severity:** Medium Risk

**Context:** [EulerEarnVault.sol#L842-L845](#), [EulerEarnVault.sol#L683-L688](#)

#### Technical Details:

The issue arises due to a missing check for `feeShares != 0`, which causes `previewRedeem` to return a higher value for `totalAssetsDepositedExpected` than the actual `totalAssetsDeposited` after harvest is executed when `feeShares == 0`.

This results in a violation of the following invariants:

- `ERC4626_REDEEM_INVARIANT_B`: `previewRedeem` MUST return close to and no more than assets redeemed at `redeem` if called in the same transaction
- `ERC4626_WITHDRAW_INVARIANT_C`: `maxWithdraw` MUST return the maximum amount of assets that could be transferred from owner through `withdraw` and not cause a revert

The following Proof of Concept (PoC) located in `CriticToFoundry.sol` demonstrates the issue:

```
function test_redeemEchidna() public {
    this.addStrategy(50511606531911687419, 0);
    this.deposit(21, 0);
    this.rebalance(0, 0, 0);
    this.donateUnderlying(12, 0);
    this.harvest();
    this.mint(6254, 0);
    _delay(101007);
    this.setPerformanceFee(1012725796);
    this.simulateYieldAccrual(987897736, 0);
    this.redeem(5557, 0);
}
```

**Impact:** Medium. The issue causes vault withdrawals to fail under specific, temporary circumstances, effectively "bricking" the vault. This can prevent integrators from being able to withdraw assets if they rely on `maxWithdraw` as a parameter.



**Recommendation:** Consider implementing the check `if (feeShares != 0)` in `_previewHarvestBeforeWithdraw`, similar to the check already implemented in `_accruePerformanceFee`.

**Developer Response:** Fixed at [PR #120](#).

# Low Risk

## L-01 - ERC4626 compliancy

**Severity:** Low Risk

**Technical Details:**

The Euler Earn vault is a yield aggregation protocol built on top of the Euler v2 vault system. It is designed around tokenised vaults (ERC4626). However, while it incorporates many aspects of the ERC4626 specification, it does not fully inherit all the invariants and properties outlined in the ERC4626PropertiesSpec. As such, certain ERC4626 compliance requirements are intentionally excluded, which may lead to deviations from the expected behavior in some edge cases.

**Recommendation:**

Document the deviations from the standard ERC4626 behavior, specifying the expected changes in property behavior. Additionally, adjust the ERC4626 invariants to align with the specific requirements and design of the Euler Earn protocol.

**Developer Response:**

Acknowledged. While the Earn vault is not intended to be fully ERC4626 compliant, certain invariants have been intentionally adjusted to better suit the protocol's requirements.

## Informational

### I-01 - Earn Vault does not execute gulp after enabling back an strategy from emergency to active

**Severity:** Informational

**Technical Details:**

The `toggleStrategyEmergencyStatus` function only triggers a gulp when the strategy status transitions from Active to Emergency. As a result, when a strategy is reverted to Active, the Earn balance is assigned to its `allocated` field, but it is not added to `totalAssetsDeposited`. This behaviour is intentional, as the balance is accounted as yield that needs to be gulped and smeared. However, the `_gulp` function is only called during the Active → Emergency transition.

**Recommendation:** Consider calling `_gulp()` at the end of the function so the entire process occurs in one transaction.

**Developer Response:** Fixed at [PR #102](#).

## 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 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, 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.