ENIGMA DARK

Securing the Shadows



Managed Security Review

# Hyperdrive Router

May, 2025

# 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

**Hyperdrive Router**
Hyperdrive is an innovative lending protocol deployed in the HyperEVM which offers various yield-strategies integrating with Hyperliquid vaults.


# Engagement Overview

Over the course of 3 days, beginning May 20 2025, the Enigma Dark team conducted a security review of the Hyperdrive Router project, specifically the contracts located at `packages/staking/contracts/protocol/router`. The review was performed by one Security Researcher: kiki.

The following repositories were reviewed at the specified commits:

| Repository | Commit |
| --- | --- |
| ambitfi/hyperdrive-contracts | `5d51505dc92bad5d964e5325b38653ef5b38bd7b` |

## 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 | 0 | 0 | 0 |
| Low | 1 | 1 | 0 |
| Informational | 3 | 2 | 1 |

# Findings

| Index | Issue Title | Status |
| --- | --- | --- |
| L-01 | Transient Storage is Not Cleared | Fixed |
| I-01 | Missing Vault Validation | Acknowledged |
| I-02 | Limited wrapETH Functionality | Fixed |
| I-03 | Missing Revert Reasons | Fixed |

# Detailed Findings

# Low Risk

## L-01 - Transient Storage is Not Cleared

**Severity**: Low Risk

**Context**:

- Router.sol#L52-L60

**Technical Details**: Whenever a user initially uses the `execute` function, it will assign the `msg.sender` as the caller; this is done via `tStore`.

Because `tStore` will persist for the entirety of the transaction, a user's second call within the same transaction to the router will still maintain the same `caller` value.

This could cause issues when integrators use the router in a way that allows arbitrary users to make calls from the third party, where the caller will be set as the integrator, and then in the same transaction make a call from their own contract. In such a case, the caller would still be the integrator while the actual `msg.sender` is the arbitrary user.

With some of the paths within `dispatch` leading to some form of transferring where the `from` address is the `caller`, if a user is able to enact these transfers while not being the caller, they could steal funds from these integrators up to an amount they have approved the router.

Currently, there is no direct way to execute this attack without being able to impersonate a market and go through the `flashloanCallback` function, which would require admin privileges.

**Impact**: Persisting state allows for potential fund loss.

**Recommendation**: At the end of the `execute` function, set `caller` to address 0. More info regarding transient storage can be read here.

**Developer Response**: Fixed at commit `24d90c9`

# Informational

## I-01 - Missing Vault Validation

**Severity**: Informational

**Context**:

- ERC4626RouterLib.sol#15
- ERC4626RouterLib.sol#34
- ERC4626RouterLib.sol#58
- ERC4626RouterLib.sol#71

**Technical Details**: When users use the router to deposit, withdraw, mint, or redeem, they need to provide a vault address. This parameter is not validated, which allows users to set any arbitrary address as the `vault` and execute any logic they desire.

**Impact**: Unexpected external calls

**Recommendation**: There is no clear impact to this, but it would still be worth considering some validation that the vault is indeed a Hyperdrive vault and not an arbitrary address.

**Developer Response**: Acknowledged. Anyone could build their own version of the router.

## I-02 - Limited wrapETH Functionality

**Severity**: Informational

**Context**: RouterLib.sol#L35

**Technical Details**: The `wrapETH` function validates that `msg.value` is equal to the amount that is being deposited. But because `msg.value` will persist through the entire `execute` flow, users will not be able to access the `wrapETH` function more than once. This may be unexpected and limit how users can use the router.

**Impact**: Limited functionality

**Recommendation**: The validation does a good job of ensuring that excess ETH is less likely to be left in the router and is doing more good than harm. Due to this, it is recommended that this behavior be documented so that users and integrators are aware.

**Developer Response**: Fixed at commit `4629536`

## I-03 - Missing Revert Reasons

**Severity**: Informational

**Context**:

- Router.sol#L120
- RouterLib.sol#L27
- RouterLib.sol#L35

**Technical Details**: In both the `Router` contract as well as the `RouterLib` library, there are `require` statements that do not contain a revert reason. Because of this, when those reverts occur, the user will not know why their transaction failed. This could lead to unneeded friction and confusion for the user.

**Impact**: Potential confusion and challenges using the Router contract.

**Recommendation**: Add a revert reason where applicable.

**Developer Response**: Fixed at commit `d5fdbc0`

# 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.