Hyperdrive Audit Report

Version 1.0

Conducted by: Kiki

March 2025

Table of Contents

1 Introduction **3**
- 1.1 About Kiki . . . 3
- 1.2 Disclaimer . . . 3
- 1.3 Risk classification . . . 3
  - 1.3.1 Impact . . . 3
  - 1.3.2 Likelihood . . . 3
  - 1.3.3 Actions required by severity level . . . 3

2 Executive Summary **4**
- 2.1 Hyperdrive . . . 4
- 2.2 Overview . . . 4
- 2.3 Scope . . . 4
- 2.4 Issues Found . . . 5
- 2.5 Findings & Resolutions . . . 5

3 Findings **6**
- 3.1 Low Risk . . . 6
  - 3.1.1 Unauthorized Logic Execution via supplyOrRefund . . . 6
  - 3.1.2 Incomplete Event Emission in Zapper Contract . . . 6

1 Introduction

1.1 About Kiki

Kiki is a Security Researcher who has conducted dozens of security reviews with the top security firm Guardian Audits, as well as through private engagements. View their previous work here, or reach out via Twitter or Telegram.

1.2 Disclaimer

Security Reviews are a time, resource, and expertise bound effort where trained experts evaluate smart contracts using a combination of automated and manual techniques to identify as many vulnerabilities as possible. Audits can show the presence of vulnerabilities **but not their absence**.

1.3 Risk classification

Severity	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

1.3.1 Impact

- **High** - leads to a significant loss of assets in the protocol or significantly harms a group of users.
- **Medium** - only a small amount of funds can be lost or a functionality of the protocol is affected.
- **Low** - any kind of unexpected behaviour that's not so critical.

1.3.2 Likelihood

- **High** - direct attack vector; the cost is relatively low to the amount of funds that can be lost.
- **Medium** - only conditionally incentivized attack vector, but still relatively likely.
- **Low** - too many or too unlikely assumptions; provides little or no incentive.

1.3.3 Actions required by severity level

- **Critical** - client **must** fix the issue.
- **High** - client **must** fix the issue.
- **Medium** - client **should** fix the issue.
- **Low** - client **could** fix the issue.

2 Executive Summary

2.1 Hyperdrive

Hyperdrive offers risk mitigated high-yield strategies to those looking to safely park their USDC. Hyperdrive is the premier DeFi hub on Hyperliquid, offering unparalleled yield strategies to all users.

More on the protocol can be found here.

2.2 Overview

Project Name	Hyperdrive
Codebase	hyperdrive-contracts, ambitfi-contracts
Operating platform	HyperEVM
Language	Solidity
Initial commit	[1] 6fefd66809e3b751d2f3518dbfa626fe4ae6aed6, [2] 465bf3bd713745d8076398e13cbdf945eed27e7b
Remediation commit	6e7ecd1c687a91eddd0cec8d30f4f048f5a11076, 465bf3bd713745d8076398e13cbdf945eed27e7b
Audit methodology	Manual Review
Engagement Duration	2 Days

2.3 Scope

Files and folders in scope

- packages/lending/contracts/protocol/zapper/Zapper.sol
- packages/core/contracts/protocol/security/AuthorizedAccessControlUpgradeable.sol
- packages/core/contracts/protocol/security/RoleBasedAccessControlUpgradable.sol
- packages/core/contracts/protocol/security/AdminAccessControlUpgradeable.sol
- packages/core/contracts/protocol/utils/SweepableUpgradeable.sol
- packages/core/contracts/protocol/utils/PausableUpgradeable.sol
- packages/core/contracts/protocol/utils/Upgradeable.sol

2.4 Issues Found

Severity	Total Found	Resolved	Partially Resolved	Acknowledged
Critical risk	0	0	0	0
High risk	0	0	0	0
Medium risk	0	0	0	0
Low risk	2	2	0	0

2.5 Findings & Resolutions

ID	Title	Severity	Status
L-01	Unauthorized Logic Execution via supplyOrRefund	Low	Resolved
L-02	Incomplete Event Emission in Zapper Contract	Low	Resolved

3 Findings

3.1 Low Risk

3.1.1 Unauthorized Logic Execution via supplyOrRefund

Severity: *Low risk (Resolved)*

Context: ***Zapper.sol***:*280-298*

Description:

The `supplyOrRefund` function in the `Zapper` contract allows arbitrary tokens to be used without validation. Since the contract doesn't validate whether `token` is a legitimate ERC20 token, a malicious user can set `tokensOut` to include their own contract that implements the ERC20 interface but contains additional malicious logic.

When the `balanceOf` function is called on this malicious contract, it can execute arbitrary code while still returning a valid balance value. This allows the attacker to disguise their contract as a token and execute unexpected logic mid-transaction. The contract could make additional calls to other Hyperdrive contracts or perform other state changes within the context of the `supplyOrRefund` function execution.

Although there's no immediate direct impact identified, this creates an unintended execution path that malicious users could potentially exploit for purposes not intended by the protocol design. The lack of token validation affects multiple functions in the `Zapper` contract since `supplyOrRefund` is called from `zapInCallback`, `zapOutCallback`, and other key functions.

Recommendation:

Implement a whitelist system for approved tokens that can be used with the `Zapper` contract. Maintain this list through admin-controlled functions that can add or remove tokens as needed. Before processing any token in the `supplyOrRefund` function, verify that it exists in the approved tokens list.

This whitelist can be expanded based on user demand, but should require administrative approval to maintain security controls over which contracts can interact with the zapper functionality.

Resolution: Resolved. The recommended fix was implemented in **6e7ecd1**.

3.1.2 Incomplete Event Emission in Zapper Contract

Severity: *Low risk (Resolved)*

Context: ***Zapper.sol***:*163-171*

Description:

The `Zapper` contract contains a bug in the `zapOut` function where only the first element of the `tokensOut` and `amountsOut` arrays is being emitted in the event log. The function correctly returns the complete array of output amounts through the `amountsOut` return value, but the `ZapOut` event only captures the first token and first amount.

This implementation is problematic because the `zapOut` function is designed to handle multiple output tokens, as evidenced by the arrays in the `ZapOutParams` structure and the `supplyOrRefund` function which processes multiple tokens. When the function processes multiple tokens, the event will only record information about the first token, causing a loss of transaction data.

This issue affects the transparency of the system, as it creates inconsistency between what's executed and what's emitted.

Recommendation:

Modify the `ZapOut` and `ZapIn` event definition in the `IZapper` interface to accept arrays of tokens and amounts rather than single values. Then update the event emission in the `Zapper` contract.

```
// In IZapper.sol
event ZapOut(
  address indexed account,
  address indexed asset,
  uint256 marketId,
  uint256 amount,
  address[] tokensOut,
  uint256[] amountsOut,
  address caller
);

// In Zapper.sol
emit ZapOut(
  msg.sender,
  params.token,
  params.marketId,
  params.amount,
  params.tokensOut,
  amountsOut,
  msg.sender
);
```

Resolution: Resolved. The recommended fix was implemented in **8ed08d8**.