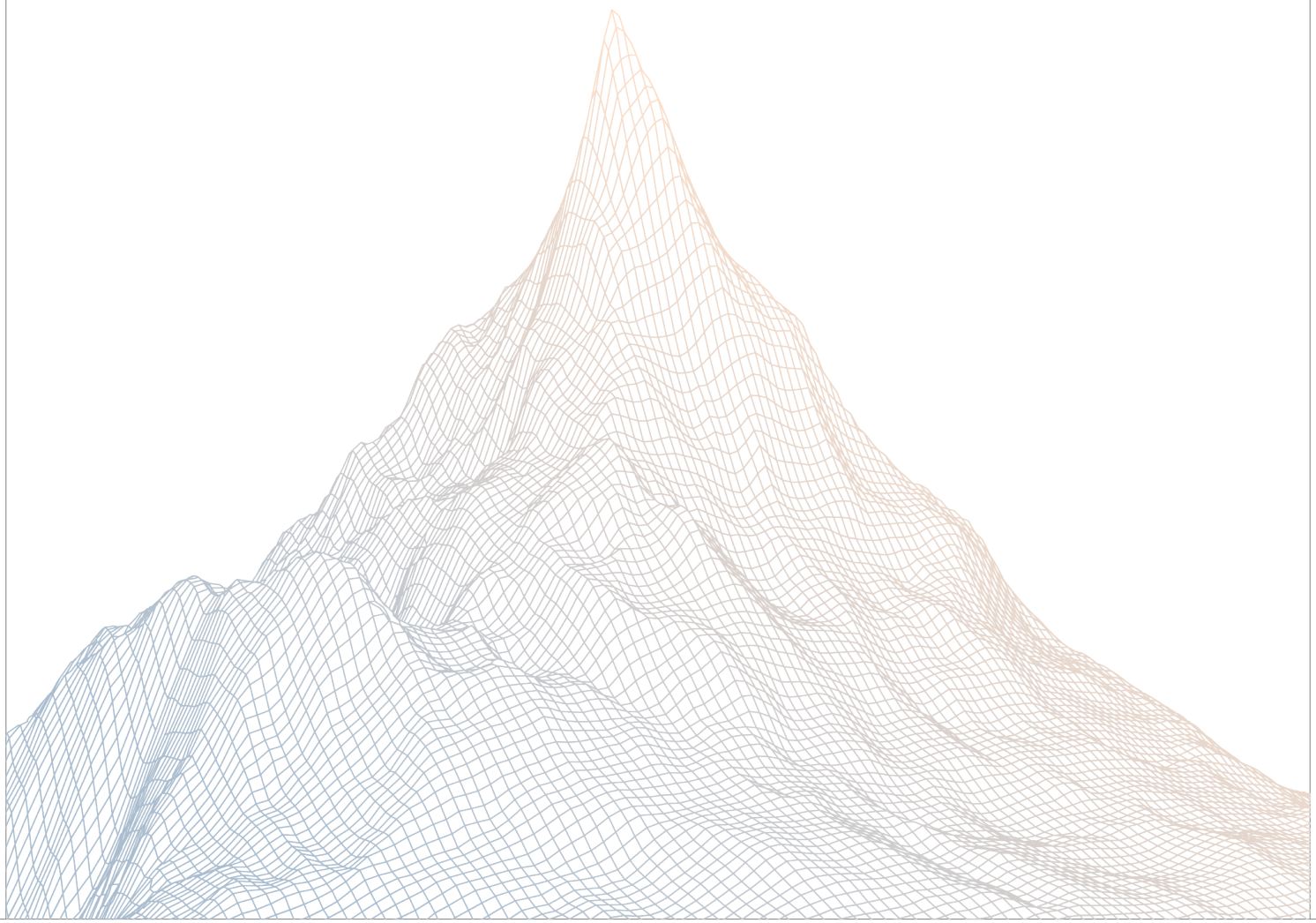
Zenith

Berachain

Smart Contract Security Assessment

VERSION 1.1



AUDIT DATES: May 29th to May 30th, 2025

AUDITED BY: ether_sky
said

Contents

1 Introduction

1.1 About Zenith

Zenith assembles auditors with proven track records: finding critical vulnerabilities in public audit competitions.

Our audits are carried out by a curated team of the industry's top-performing security researchers, selected for your specific codebase, security needs, and budget.

Learn more about us at https://zenith.security.

1.2 Disclaimer

This report reflects an analysis conducted within a defined scope and time frame, based on provided materials and documentation. It does not encompass all possible vulnerabilities and should not be considered exhaustive.

The review and accompanying report are presented on an "as-is" and "as-available" basis, without any express or implied warranties.

Furthermore, this report neither endorses any specific project or team nor assures the complete security of the project.

1.3 Risk Classification

SEVERITY LEVEL	IMPACT: HIGH	IMPACT: MEDIUM	IMPACT: LOW
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2

Executive Summary

2.1 About Berachain

Berachain is a high-performance EVM-Identical Layer 1 blockchain utilizing Proof-of-Liquidity (PoL) and built on top of the modular EVM-focused consensus client framework BeaconKit.

2.2 Scope

The engagement involved a review of the following targets:

Target	meta-aggregator
Repository	https://github.com/berachain/meta-aggregator
Commit Hash	`75c3cdb62af832c0211de3f1b378669e89951c10`
Files	`src/MetaAggregator.sol`

2.3 Audit Timeline

May 29, 2025	Audit start
May 30, 2025	Audit end
June 2, 2025	Report published

2.4 Issues Found

SEVERITY	COUNT
Critical Risk	0
High Risk	0
Medium Risk	1
Low Risk	1
Informational	0
Total Issues	**2**

3

Findings Summary

ID	Description	Status
M-1	The swapFee and affiliateFee should be set to zero when using the Magpie Router	Acknowledged
L-1	When fromAsset is not a native token, the native token amount should be zero	Resolved

4

Findings

4.1 Medium Risk

A total of 1 medium risk findings were identified.

[M-1] The swapFee and affiliateFee should be set to zero when using the Magpie Router

SEVERITY: Medium	IMPACT: Medium
STATUS: Acknowledged	LIKELIHOOD: Low

Target

- MetaAggregator.sol

Description:

When using the `Magpie Router`, only the specified `amount` of the `input token` is approved for the `router` to use.

- MetaAggregator.sol#L200

```
function _handleERC20(
    IERC20 inputToken,
    address aggregator,
    uint256 amountIn,
    uint256 swapFeeAmount
)
    private
{
    inputToken.safeTransferFrom(msg.sender, address(this), amountIn
    + swapFeeAmount);
    _handleSwapFee(inputToken, swapFeeAmount);
@->     inputToken.forceApprove(aggregator, amountIn);
}
```

In the case of a `native token`, the same behavior applies, as the `input amount` is sent directly to the `router`.

- MetaAggregator.sol#L164

```
function _handleSwap(
    address inputToken,
    address aggregator,
    uint256 amountIn
)
    private
    returns (uint256 nativeTokenAmount)
{
    uint256 swapFeeAmount = _getSwapFee(amountIn);
    if (inputToken == NATIVE_TOKEN) {
@->          nativeTokenAmount = amountIn;
        _handleNativeToken(amountIn, swapFeeAmount);
    } else {
        _handleERC20(IERC20(inputToken), aggregator, amountIn,
    swapFeeAmount);
    }
}
```

However, if `swapFee` and `affiliateFee` are not set to 0, these additional amounts must be accounted for. 0x52bebb9706...

```
function swap(
    SwapData memory swapData,
    address fromAddress,
    uint256 fullAmountIn
) private returns (uint256 amountOut, uint256 gasUsed) {
    if (swapData.fromAssetAddress.isNative()) {
@->          if (msg.value < (swapData.amountIn + swapData.swapFee
    + swapData.affiliateFee)) {
            revert InvalidAmountIn();
        }
    }
}
```

So the transaction will revert due to insufficient funds.

Recommendations:

Add a check to ensure that `swapFee` and `affiliateFee` are zero, or include these amounts in the total input calculation.

Berachain: Acknowledged. Our use-case never require us to charge affiliateFee on swap and swapFee on magpie is only for swapWithUserSignature where users sign the request and it gets executed by magpie hence they take swap fees on such swap.

In our case, frontend will send callData accordingly and therefore we don't see any further need to enforce these checks in order to keep it gas efficient.

Zenith: Acknowledged.

4.2 Low Risk

A total of 1 low risk findings were identified.

[L-1] When fromAsset is not a native token, the native token amount should be zero

SEVERITY: Low	IMPACT: Low
STATUS: Resolved	LIKELIHOOD: Low

Target

- MetaAggregator.sol

Description:

There is no check to ensure that `msg.value` is zero when `fromAsset` is not a `native token`.

- MetaAggregator.sol#L117-L125

```
function swapWithMagpie(bytes calldata swapArgs)
    external payable whenNotPaused returns (uint256 amountOut) {
    SwapData memory swapData = LibRouter.getData();
    uint256 amountIn = swapData.amountIn;
    address fromAssetAddress = swapData.fromAssetAddress;
    // handle input token for swap and charge the swap fee, nativeTokenAmount
    is 0 in case of non-native token.
    uint256 nativeTokenAmount = _handleSwap(fromAssetAddress, magpieRouter,
    amountIn);
    amountOut = IMagpieRouterV3_1(magpieRouter).swapWithMagpieSignature{
    value: nativeTokenAmount }(swapArgs);
    emit SwapWithMagpie(fromAssetAddress, amountIn, amountOut);
}
```

If users accidentally send `native tokens` along with the `fromAsset` input, those `native tokens` cannot be recovered

Recommendations:

```
function swapWithMagpie(bytes calldata swapArgs)
    external payable whenNotPaused returns (uint256 amountOut) {
    SwapData memory swapData = LibRouter.getData();
    uint256 amountIn = swapData.amountIn;
    address fromAssetAddress = swapData.fromAssetAddress;
+^^Irequire(fromAssetAddress == NATIVE_TOKEN || msg.value == 0, "");
    // handle input token for swap and charge the swap fee, nativeTokenAmount
    is 0 in case of non-native token.
    uint256 nativeTokenAmount = _handleSwap(fromAssetAddress, magpieRouter,
    amountIn);
    amountOut = IMagpieRouterV3_1(magpieRouter).swapWithMagpieSignature{
    value: nativeTokenAmount }(swapArgs);
    emit SwapWithMagpie(fromAssetAddress, amountIn, amountOut);
}

function swapWithOB(
    IOBRouter.swapTokenInfo memory tokenInfo,
    bytes calldata pathDefinition,
    address executor
)
    external
    payable
    whenNotPaused
    returns (uint256 amountOut)
{
    // cache input amount and input token
    uint256 inputAmount = tokenInfo.inputAmount;
    address inputToken = tokenInfo.inputToken;
+^^Irequire(inputToken == NATIVE_TOKEN || msg.value == 0, "");
    // handle input token for swap and charge the swap fee, nativeTokenAmount
    is 0 in case of non-native token.
    uint256 nativeTokenAmount = _handleSwap(inputToken, obRouter,
    inputAmount);
    amountOut = IOBRouter(obRouter).swap{ value: nativeTokenAmount
    }(tokenInfo, pathDefinition, executor, 0);
    emit SwapWithOB(inputToken, inputAmount, amountOut);
}
```

Berachain: Fixed in the commit PR-2

Zenith: Verified.