decurity

Smart Contract Security Audit Report

Gearbox Balancer Integration Audit

1. Contents

1. Contents 2
2. General Information 3
2.1. Introduction 3
2.2. Scope of Work 3
2.3. Threat Model 3
2.4. Weakness Scoring 4
2.5. Disclaimer 4
3. Summary 5
3.1. Suggestions 5
4. General Recommendations 6
4.1. Security Process Improvement 6
5. Findings 7
5.1. Typo in error message 7
5.2. Inconsistent version constants across Balancer V3 integration contracts 7
5.3. wethIsEth may cause funds lost 8
6. Appendix 12
6.1. About us 12

2. General Information

This report contains information about the results of the security audit of the (hereafter referred to as “Customer”) smart contracts, conducted by Decurity in the period from 2025-10-17 to 2025-10-21.

2.1. Introduction

Tasks solved during the work are:

- Review the protocol design and the usage of 3rd party dependencies,
- Audit the contracts implementation,
- Develop the recommendations and suggestions to improve the security of the contracts.

2.2. Scope of Work

The audit scope included the contracts in the following repository: Gearbox-protocol/integrations-v3/. Initial review was done for the commit 44dfdca1edba59d0183a4cf0416712d3ee1e847e and the re-testing was done for the commit XXX.

The following contracts have been tested:

- contracts/helpers/balancer/BalancerV3RouterGateway.sol
- contracts/adapters/balancer/BalancerV3WrapperAdapter.sol
- contracts/adapters/balancer/BalancerV3RouterAdapter.sol

2.3. Threat Model

The assessment presumes actions of an intruder who might have capabilities of any role (an external user, token owner, token service owner, a contract). The centralization risks have not been considered upon the request of the Customer.

The main possible threat actors are:

- User,
- Protocol owner,

- Liquidity Token owner/contract.

2.4. Weakness Scoring

An expert evaluation scores the findings in this report, an impact of each vulnerability is calculated based on its ease of exploitation (based on the industry practice and our experience) and severity (for the considered threats).

2.5. Disclaimer

Due to the intrinsic nature of the software and vulnerabilities and the changing threat landscape, it cannot be generally guaranteed that a certain security property of a program holds.

Therefore, this report is provided “as is” and is not a guarantee that the analyzed system does not contain any other security weaknesses or vulnerabilities. Furthermore, this report is not an endorsement of the Customer’s project, nor is it an investment advice.

That being said, Decurity exercises best effort to perform their contractual obligations and follow the industry methodologies to discover as many weaknesses as possible and maximize the audit coverage using the limited resources.

3. Summary

As a result of this work, we did not identify any critical, exploitable security issues.

The other suggestions included best practices.

The team has given the feedback for the suggested changes and explanation for the underlying code.

3.1. Suggestions

The table below contains the discovered issues, their risk level, and their status as of October 21, 2025.

Table. Discovered weaknesses

Issue	Contract	Risk Level	Status
Typo in error message	contracts/helpers/balancer/BalancerV3RouterGateway.sol	**Info**	Not fixed
Inconsistent version constants across Balancer V3 integration contracts	contracts/adapters/balancer/BalancerV3WrapperAdapter.sol	**Info**	Not fixed
wethIsEth may cause funds lost	contracts/helpers/balancer/BalancerV3RouterGateway.sol	**Info**	Not fixed

4. General Recommendations

This section contains general recommendations on how to improve overall security level.

The Findings section contains technical recommendations for each discovered issue.

4.1. Security Process Improvement

The following is a brief long-term action plan to mitigate further weaknesses and bring the product security to a higher level:

- Keep the whitepaper and documentation updated to make it consistent with the implementation and the intended use cases of the system,
- Perform regular audits for all the new contracts and updates,
- Ensure the secure off-chain storage and processing of the credentials (e.g. the privileged private keys),
- Launch a public bug bounty campaign for the contracts.

5. Findings

5.1. Typo in error message

Risk Level: Info

Status: Not fixed

Contracts:

- contracts/helpers/balancer/BalancerV3RouterGateway.sol

Location:

- `addLiquidityUnbalanced()`

Description:

The error message at line 87 contains a typo where "BalanceV3RouterGateway" is missing the letter 'r' in "Balancer". This makes the error message inconsistent with the actual contract name and potentially confusing during debugging.

```
if (exactAmountsIn.length != tokens.length)
revert("BalanceV3RouterGateway: amounts array length mismatch");
```

The error string says "BalanceV3RouterGateway" but should say "BalancerV3RouterGateway" to match the contract name.

Remediation:

Update the error message to correct the typo.

5.2. Inconsistent version constants across Balancer V3 integration contracts

Risk Level: Info

Status: Not fixed

Contracts:

- contracts/adapters/balancer/BalancerV3WrapperAdapter.sol

Description:

The `BalancerV3WrapperAdapter` contract declares version `3_10`, while the related `BalancerV3RouterAdapter` contract (deployed as part of the same Balancer V3 integration) uses version `3_11`.

```
// BalancerV3WrapperAdapter.sol:20
uint256 public constant override version = 3_10;
 // BalancerV3RouterAdapter.sol:27
uint256 public constant override version = 3_11;
```

Since these contracts are part of the same integration suite and likely deployed together, they should maintain consistent version numbering for clarity and proper version management.

Remediation:

Update the version constant in `BalancerV3WrapperAdapter.sol` to match the other Balancer V3 integration contracts.

5.3. wethIsEth may cause funds lost

Risk Level: Info

Status: Not fixed

Contracts:

- contracts/helpers/balancer/BalancerV3RouterGateway.sol

Location:

- `swapSingleTokenExactIn()`
- `removeLiquiditySingleTokenExactIn()`

Description:

There is currently no way to withdraw ETH from the contract. ETH can end up in the contract either because Balancer's router unwraps WETH to ETH when `wethIsEth` is set to true, or because ETH is sent back by the router. The receive() function accepts ETH, and the comment states that such ETH is intended to be unrecoverable.

```
/// @dev The receive function is required in case Balancer sends back
ETH. It is intended for received ETH to be unrecoverable. // @audit
eth can be transfered from weth
    receive() external payable {}
```

In both `swapSingleTokenExactIn` and `removeLiquiditySingleTokenExactIn`, the `wethIsEth` flag is forwarded to the router. If `tokenOut` is WETH and `wethIsEth == true`, the router may unwrap and deliver ETH to this contract. Since the contract has no ETH recovery or auto-wrapping path, those funds become stuck. The `_transferBalance(tokenOut)` only forwards ERC-20 balances.

```
 function swapSingleTokenExactIn(
        address pool,
        IERC20 tokenIn,
        IERC20 tokenOut,
        uint256 exactAmountIn,
        uint256 minAmountOut,
        uint256 deadline,
        bool wethIsEth,
        bytes calldata userData
    ) external returns (uint256 amountOut) {
        exactAmountIn = _transferTokenIn(tokenIn, exactAmountIn);
         tokenIn.forceApprove(permit2, exactAmountIn);
        IPermit2(permit2).approve(address(tokenIn), balancerV3Router,
uint160(exactAmountIn), uint48(block.timestamp));
         IBalancerV3Router(balancerV3Router).swapSingleTokenExactIn(
            pool, tokenIn, tokenOut, exactAmountIn, minAmountOut, deadline,
wethIsEth, userData // @audit set wethIsEth as false
        );
         amountOut = _transferBalance(tokenOut);
         tokenIn.forceApprove(permit2, 1);
         return amountOut;
    }
```

```
function removeLiquiditySingleTokenExactIn(
        address pool,
        uint256 exactBptAmountIn,
        IERC20 tokenOut,
        uint256 minAmountOut,
        bool wethIsEth,
        bytes memory userData
    ) external returns (uint256 amountOut) {
        exactBptAmountIn = _transferTokenIn(IERC20(pool), exactBptAmountIn);
        minAmountOut = minAmountOut == 0 ? 1 : minAmountOut;
         IERC20(pool).forceApprove(balancerV3Router, exactBptAmountIn);
         IBalancerV3Router(balancerV3Router).removeLiquiditySingleTokenExact
In(
            pool, exactBptAmountIn, tokenOut, minAmountOut, wethIsEth,
userData // @audit setting wethIsEth causes user funds, set wethIsEth as
false
        );
         amountOut = _transferBalance(tokenOut);
         IERC20(pool).forceApprove(balancerV3Router, 1);
    }
```

While BalancerV3RouterAdapter always sets `wethIsEth` to false, other callers may use BalancerV3RouterGateway with `tokenOut == WETH` and `wethIsEth == true`, causing ETH to be sent here and trapped.

Remediation:

Consider setting `wethIsEth` as false and adding the recovery function.

References:

- https://github.com/balancer/balancer-v3-monorepo/blob/2f0eb1845eb7d40c83904b669cddf37c733479d3/pkg/vault/contracts/RouterCommon.sol#L292

6. Appendix

6.1. About us

The Decurity team consists of experienced hackers who have been doing application security assessments and penetration testing for over a decade.

During the recent years, we've gained expertise in the blockchain field and have conducted numerous audits for both centralized and decentralized projects: exchanges, protocols, and blockchain nodes.

Our efforts have helped to protect hundreds of millions of dollars and make web3 a safer place.