OpenZeppelin | security

Beefy Zap Audit

June 27, 2023

This security assessment was prepared by OpenZeppelin.

Table of Contents

Summary

Type	DeFi
Timeline	From 2023-06-05 To 2023-06-08
Languages	Solidity

Total Issues	12 (12 resolved)
Critical Severity Issues	1 (1 resolved)
High Severity Issues	1 (1 resolved)
Medium Severity Issues	0 (0 resolved)
Low Severity Issues	4 (4 resolved)
Notes & Additional Information	6 (6 resolved)

Scope

We audited the beefy-zap repository at the addd5741a520b10924f1f26cc45208ff5fa88139 commit.

In scope were the following contracts:

```
contracts
├── interfaces
│   ├── IBeefyTokenManager.sol
│   ├── IBeefyZapRouter.sol
│   └── IPermit2.sol
└── zaps
    ├── BeefyTokenManager.sol
    └── BeefyZapRouter.sol
    └── ZapErrors.sol
```

System Overview

`BeefyZapRouter` serves as a versatile intermediary designed to execute users' orders through routes supplied by the users or by a third party. The user has the option to independently execute an order by either selecting the desired route or alternatively signing a permit with `Permit2` which enables third parties to carry out the order on their behalf.

User Order Execution

In order to execute the order independently, the user is required to approve the `BeefyTokenManager` contract as a spender of the desired input tokens. The user can then proceed to execute the order through the `BeefyZapRouter` contract. This contract will retrieve the user's tokens via the `BeefyTokenManager` and carry out the order based on the provided route.

Third-Party Order Execution

The protocol enables users to delegate the execution of orders to third parties by leveraging the `Permit2` protocol for managing user approvals. To enable the execution of orders on behalf of users, it is necessary for them to approve spending by the `Permit2` contract. They can then sign an order including the input and minimum output token amounts, which allows any third party to execute it by supplying a route.

Security Model & Trust Assumptions

`BeefyZapRouter` heavily relies on `Permit2` to allow users to interact with the protocol via off-chain signed orders. The `Permit2` contract is trusted to behave according to its specification.

Privileged Roles

The owner of the `BeefyZapRouter` can pause and unpause the contract which will prevent it from executing users' orders.

Critical Severity

C-01 External Call to `Permit2` Allows Stealing Users' Tokens

During an order execution, it is possible to make arbitrary external calls to any `stepTarget` except the token manager. A vulnerability arises when setting `stepTarget` equal to the `Permit2` address, which allows an attacker to make a call from the router to `Permit2` using a valid order/signature pair from another benign user.

This results in the transfer of all tokens from the benign user to the router during the attacker's order execution, enabling them to steal all the tokens from the order.

Consider disallowing any external calls to `Permit2` in both steps and relaying executions.

Update: *Resolved in pull request #6 at commit 6ba7a7e. External calls to the `Permit2` contract have been prohibited.*

High Severity

H-01 Denial-of-Service of Router Functionality

The `_approveToken` function of the `BeefyZapRouter` contract is responsible for approving each `stepTarget` to spend an unlimited amount of tokens. The function checks if the allowance is lower than the requested amount, and in that case it sets the approval to the maximum value.

However, when approving non-zero amounts, `safeApprove` (which is used in this function) requires the current allowance to be equal to zero.

This requirement can be manipulated by making use of external calls in stages or relays. These calls can directly communicate with one of the accepted tokens, setting a low allowance for the exchange. Consequently, this causes future attempts to utilize the token on the targeted exchange to revert.

Consider replacing the usage of `safeApprove` with the newer `forceApprove` function from the OpenZeppelin library.

Update: *Resolved in pull request #7 at commit 76fd619. The usage of `safeApprove` in `_approveToken` has been replaced with the new `forceApprove` function.*

Low Severity

L-01 Floating Pragma

Throughout the codebase, the version of Solidity used is `^0.8.0`. This version indicates that any compiler version after `0.8.0` can be used to compile the source code. However, the code will not compile when using version `0.8.3` or earlier since there are functions that use custom errors that were introduced in Solidity `0.8.4`.

Consider upgrading all contracts to Solidity version `0.8.4` at a minimum, but ideally to the latest version. This precautionary measure also helps prevent the accidental deployment of contracts with outdated compiler versions that could potentially introduce vulnerabilities.

Update: *Resolved in pull request #8 at commit 71c3eee. The Solidity version has been locked to `0.8.19`.*

L-02 Missing Docstrings

Throughout the codebase, there are several parts that do not have docstrings. For instance:

- Line 8 of `BeefyTokenManager.sol`
- Line 5 of `ZapErrors.sol`

When writing docstrings, consider following the Ethereum Natural Specification Format (NatSpec).

Update: *Resolved in pull request #9 at commit 132902c. The docstrings have been added to all contracts and interfaces.*

L-03 Witness Type String Does Not Follow EIP-712

The `BeefyZapRouter` contract does not implement EIP-712 correctly for `Permit2`'s `permitWitnessTransferFrom` functionality. The witness string used by the `BeefyZapRouter` does not follow `Uniswap`'s guidance on integrating with `Permit2` and results in an incorrect EIP-712 type string being created.

This string is missing the declaration of `Order order` inside `PermitBatchWitnessTransferFrom` after the `deadline` parameter. In addition, there is no definition of the `TokenPermissions` object.

Consider updating the `ORDER_STRING` value to include the `Order order` parameter and the `TokenPermissions` object.

Update: *Resolved in pull request #10 at commit e2aad12.*

L-04 Missing `Order` Typehash Prefix for Witness Parameter

The `witness` parameter used by `BeefyZapRouter` does not follow Uniswap's guidance on integrating with `Permit2` and is calculated only out of `Order` data, instead of prefixing the data with the `Order` typehash.

Consider prefixing `Order` data with the typehash before executing the `keccak256` function.

Update: *Resolved in pull request #11 at commit 0bcdf0c. The `Order` typehash has been added to the `witness` parameter.*

Notes & Additional Information

N-01 Styling Suggestions

In the `IBeefyZapRouter.sol` contract, external files are imported before declaring the Solidity compiler version.

Consider declaring the Solidity version first to comply with the Solidity style guide.

Update: *Resolved in pull request #12 at commit 0a5fc06. The order of the layout has been changed by declaring the Solidity version first and then importing external files.*

N-02 Missing Custom Errors

The `BeefyTokenManager` contract is using a `require` statement to validate the caller of the `pullTokens` function. Since solidity version `0.8.4`, custom errors provide a cleaner and more cost-efficient way to explain to users why an operation failed.

To improve the clarity of the codebase and save gas, consider replacing the error string with a custom error.

Update: *Resolved in pull request #13 at commit 059af74. The `require` statement in the `pullTokens` function has been replaced with the custom error `CallerNotZap`.*

N-03 Typographical Errors

Throughout the codebase, the following typographical errors have been identified:

- Line 16: "TokenManger" should be "TokenManager".
- Line 147: "appoved" should be "approved".

Consider fixing these typographical errors to improve the readability of the codebase.

Update: *Resolved in pull request #14 at commit 233fb84.*

N-04 Unused Import

In `IBeefyZapRouter.sol`, the import `IBeefyTokenManager` is unused and could be removed.

Consider removing unused imports to improve the overall clarity and readability of the codebase.

Update: *Resolved in pull request #15 at commit 606ea46. The unused `IBeefyTokenManager` import has been moved from the `IBeefyZapRouter` interface to the `BeefyZapRouter` contract.*

N-05 Non-Explicit Imports Are Used

The use of non-explicit imports in the codebase can decrease the clarity of the code, and may create naming conflicts between locally defined and imported variables. This is particularly relevant when multiple contracts exist within the same Solidity files or when inheritance chains are long.

Throughout the codebase, global imports are being used. For instance:

- Line 5 of `IBeefyTokenManager.sol`
- Line 3 of `IBeefyZapRouter.sol`
- Line 4 of `IBeefyZapRouter.sol`
- Line 5 of `BeefyTokenManager.sol`
- Line 6 of `BeefyTokenManager.sol`
- Line 5 of `BeefyZapRouter.sol`
- Line 6 of `BeefyZapRouter.sol`
- Line 7 of `BeefyZapRouter.sol`
- Line 8 of `BeefyZapRouter.sol`
- Line 10 of `BeefyZapRouter.sol`
- Line 11 of `BeefyZapRouter.sol`

Following the principle that clearer code is better code, consider using named import syntax (`import {A, B, C} from "X"`) to explicitly declare the imported contracts.

Update: *Resolved in pull request #16 at commit 151dd3d.*

N-06 Interface Mismatch

Consider matching the `BeefyZapRouter` contract implementation by correctly marking the following functions as `view` in the `IBeefyZapRouter` interface:

- permit2
- tokenManager

Update: *Resolved in pull request #17 at commit c5ddf8c.*

Conclusions

One critical and one high-severity issues were discovered among various lower-severity issues. Communication with the team was smooth as they openly expressed their considerations regarding the design. Furthermore, they promptly initiated the implementation of suggested solutions to address the vulnerabilities that were disclosed early on during the audit.

Regarding the future integration of the router with the broader protocol, its definitive impact on the system's security will necessitate careful review upon completion. Since calling the router from another contract introduces arbitrary external calls, examining the router alone cannot guarantee protection against a potential attack on another contract that calls it.

Appendix

Monitoring Recommendations

While the audit helps identify code-level issues in the current implementation, the Beefy team is encouraged to incorporate monitoring activities for deployed contracts. Specifically, it is recommended to monitor the `TokenReturned` event and ensure that the amounts of returned tokens are equal to or greater than the minimum outputs expected in the signed orders.