CERTIK

Beefy Contracts

Security Assessment

Mar 5th, 2021

For :

Beefy Finance

Disclaimer

What is a CertiK report?

- A document describing in detail an in depth analysis of a particular piece(s) of source code provided to CertiK by a Client.
- An organized collection of testing results, analysis and inferences made about the structure, implementation and overall best practices of a particular piece of source code.
- Representation that a Client of CertiK has indeed completed a round of auditing with the intention to increase the quality of the company/product’s IT infrastructure and or source code.

Overview

Project Summary

Project Name	Beefy Finance
Description	DeFi
Platform	Binance Smart Chain; Solidity
Codebase	GitHub Repository
Commit	80b2123a62a5536d68208e0cd44ba4c47efaba2f, 4a1f0ba1b7302154a7f029d8956df3ae077863e0, 0ac1401c3740b7854bf4116fdf5af545d99d2baa, 6b33479dfda162987c18e09601e8beb405a06d8d, 8259e6a4382f228321cf130160a1763e0a6b6ad8

Audit Summary

Delivery Date	Mar. 5th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Feb. 22th, 2021 - Mar. 5th, 2021

Vulnerability Summary

Total Issues	14
Total Critical	0
Total Major	2
Total Minor	1
Total Informational	11

Executive Summary

This report has been prepared for **Beefy** smart contract to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Dynamic Analysis, Static Analysis, and Manual Review techniques.

The auditing process pays special attention to the following considerations:

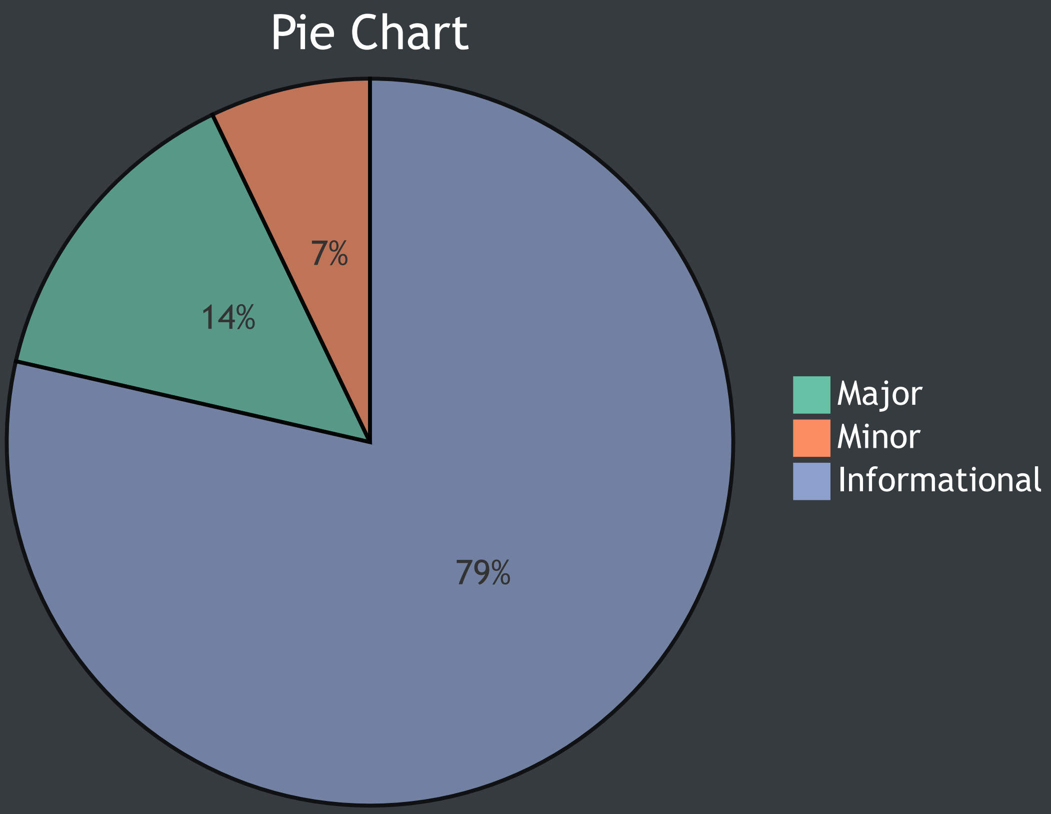
- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Below contracts:
`0xbb4CdB9CBd36B01bD1cBaEBF2De08d9173bc095c`, `0x0E09FaBB73Bd3Ade0a17ECC321fD13a19e81cE82`, `0xCa3F508B8e4Dd382eE878A314789373D80A5190A`, `0x05fF2B0DB69458A0750badebc4f9e13aDd608C7F`, `0x73feaa1eE314F8c655E354234017bE2193C9E24E`, `0x453D4Ba9a2D594314DF88564248497F7D74d6b2C`, `0x4A32De8c248533C28904b24B4cFCFE18E9F2ad01`
which are injected in the project contracts, are not in the scope of this audit. We assume these contracts are valid and non-vulnerable actors.

File in Scope

ID	Contract	SHA-256 Checksum
SC	StrategyCake.sol	7845f2ad103e2671a8156aba6cc773cc4270ba16dea088b7ec9fc995804b42af
SS	StrategySyrup.sol	fdaaa945ea47ab68dd85ebac6512503350ca90c18c92aa460d3618e1b3fbc0b9
YB	YieldBalancer.sol	9d5c3078d54d29ebab295a3800e1cbf335c7029e7c7661754f9dd208c54fd0e7
BV	BeefyVaultV4.sol	85f0efdc33cdbcee73c9c70997f288e6fc56ec72559edc3600fbb45a536dda19

Findings



ID	Title	Type	Severity	Resolved
SC-01	Ignored return values	Control Flow	Informational	❗✓
SC-02	Dangerous usage of `tx.origin`	Logic Issue	Informational	❗
SC-03	Corner case for non-contract caller check	Volatile Code	Informational	❗✓
SC-04	Proper usage of "public" and "external" types	Optimization	Informational	✓
SS-01	Ignored return values	Control Flow	Informational	❗✓
SS-02	Dangerous usage of `tx.origin`	Logic Issue	Informational	❗
SS-03	Corner case for non-contract caller check	Volatile Code	Informational	❗✓
SS-04	Proper usage of "public" and "external" types	Optimization	Informational	✓
YB-01	Function `balanceOfWant` is reusable	Optimization	Informational	✓
YB-02	Ignored return value of tranfer function	Control Flow	Minor	✓
YB-03	Lack of Reentrancy check for mutex	Control Flow	Major	✓
BV-01	Function `available` is reusable	Optimization	Informational	❗
BV-02	Lack of Reentrancy check for mutex	Control Flow	Major	✓
BV-03	Proper usage of "public" and "external" types	Optimization	Informational	✓

SC-01: Ignored return values

Type	Severity	Location
Control Flow	Informational	StrategyCake.sol [L160, L169 and L208]

Description:

`swapExactTokensForTokens` and `transfer` are not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of `swapExactTokensForTokens` and `transfer` before continuing processing.

Alleviation:

The development team replaced `transfer` by `safeTransfer` in commit 6b33479dfda162987c18e09601e8beb405a06d8d.

SC-02: Dangerous usage of `tx.origin`

Type	Severity	Location
Logic Issue	Discussion	StrategyCake.sol [L131]

Description:

In an extreme case, if the contract owner calls some malicious contract A, and within A the calling stack is like: A -> ... -> `masterchef` -> ... -> `vault` -> `StrategyCake.withdraw` , there is a chance that some unexpected behavior would happen. Such potential issue should be within a controllable range, considering the working flow after the owner checking, and also we assume `vault` is under control of project Beefy. But we still want to point out project client should be careful about using `tx.origin` .

Recommendation:

We recommend the contract owner should be careful and avoid calling any uncertain contract.

Alleviation:

No alleviation.

SC-03: Corner case for non-contract caller check

Type	Severity	Location
Volatile Code	Informational	StrategyCake.sol [L140]

Description:

`Address.isContract` cannot 100% guarantee the caller is a non-contract user, since EXTCODESIZE returns 0 if it is called from the constructor of another contract. Please consider if this is a problem for the project.

Alleviation:

The development team removed call to `Address.isContract` in commit 6b33479dfda162987c18e09601e8beb405a06d8d.
(**Beefy Finance Team - Response**)
We should be okay being called by contracts as we're protected from arbitrage by the withdrawal fees.

SC-04: Proper usage of "public" and "external" types

Type	Severity	Location
Optimization	Informational	StrategyCake.sol [L179]

Description:

Public function `balanceOf` is never called within the contract, so it could be declared as `external` .

Recommendation:

We recommend changing `public` to `external` for lower execution gas cost.

Alleviation:

The development team heeded our advice and resolved this issue in commit 6b33479dfda162987c18e09601e8beb405a06d8d.

SS-01: Ignored return values

Type	Severity	Location
Control Flow	Informational	StrategySyrup.sol [L170, L179, L190, L226]

Description:

`swapExactTokensForTokens` and `transfer` are not void-returning functions. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of `swapExactTokensForTokens` and `transfer` before continuing processing.

Alleviation:

The development team replaced `transfer` by `safeTransfer` in commit 8259e6a4382f228321cf130160a1763e0a6b6ad8.

SS-02: Dangerous usage of `tx.origin`

Type	Severity	Location
Logic Issue	Discussion	StrategySyrup.sol [L139]

Description:

In an extreme case, the contract owner calls some malicious contract A, and within A the calling stack is like: A -> ... -> `masterchef` -> ... -> `vault` -> `StrategyCake.withdraw` , there is a chance that some unexpected behavior would happen. The potential issue should be within a controllable range, considering the working flow after the owner checking in the contract, and also we assume the `vault` is under control of project Beefy. But we still want to point out that project client should be careful about using `tx.origin` .

Recommendation:

We recommend the contract owner should be careful and avoid calling any uncertain contract.

Alleviation:

No alleviation.

SS-03: Corner case for non-contract caller check

Type	Severity	Location
Volatile Code	Informational	StrategySyrup.sol [L155]

Description:

`Address.isContract` cannot 100% guarantee the caller is a non-contract user, since EXTCODESIZE returns 0 if it is called from the constructor of another contract. Please consider if this is a problem for the project.

Alleviation:

The development team removed call to `Address.isContract` in commit 6b33479dfda162987c18e09601e8beb405a06d8d.
(**Beefy Finance Team - Response**)
We should be okay being called by contracts as we're protected from arbitrage by the withdrawal fees.

SS-04: Proper usage of "public" and "external" types

Type	Severity	Location
Optimization	Informational	StrategySyrup.sol [L197]

Description:

Public function `balanceOf` is never called within the contract, so it could be declared as `external`.

Recommendation:

We recommend changing `public` to `external` for lower execution gas cost.

Alleviation:

The development team heeded our advice and resolved this issue in commit 8259e6a4382f228321cf130160a1763e0a6b6ad8.

YB-01: Function `balanceOfWant` is reusable

Type	Severity	Location
Optimization	Informational	YieldBalancer.sol [L134, L141, L149, L191, L355, L424]

Description:

The contract already maintains a wrapper function to calculate `IERC20(want).balanceOf(address(this))` , but the same verbose code repeated in different places.

Recommendation:

We recommend reusing the function `balanceOfWant` for better code maintainability and readability. Otherwise the function `balanceOfWant` should be declared as `external` instead of `public` .

Alleviation:

The development team heeded our advice and resolved this issue in commit 0ac1401c3740b7854bf4116fdf5af545d99d2baa.

YB-02: Ignored return value of tranfer function

Type	Severity	Location
Control Flow	Minor	YieldBalancer.sol [L425]

Description:

`transfer` is not a void-returning function. Ignoring the return value might cause some unexpected exception, especially if the callee function doesn't revert automatically when failing.

Recommendation:

We recommend checking the output of `transfer` before continuing processing.

Alleviation:

The development team replaced `transfer` by `safeTransfer` in commit 0ac1401c3740b7854bf4116fdf5af545d99d2baa.

YB-03: Lack of Reentrancy check for mutex

Type	Severity	Location
Control Flow	Major	YieldBalancer.sol [L289]

Description:

Function `deleteWorker` lacks check for mutex against reentrancy attack.

Recommendation:

We recommend using OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier.

Alleviation:

The development team heeded our advice and resolved this issue in commit 0ac1401c3740b7854bf4116fdf5af545d99d2baa.

BV-01: Function `available` is reusable

Type	Severity	Location
Optimization	Informational	BeefyVaultV4.sol [L72, L106, L108, L147, L151]

Description:

The contract already maintains a wrapper function to calculate `token.balanceOf(address(this))` , but the same verbose code repeated in different places.

Recommendation:

We recommend reusing the function `available` (as L126) for better code maintainability and readability.

Alleviation:

No alleviation.

BV-02: Lack of Reentrancy check for mutex

Type	Severity	Location
Control Flow	Major	BeefyVaultV4.sol [L104, L143, L180]

Description:

Function `deposit` , `withdraw` and `upgradeStrat` lack check for mutex against reentrancy attack.

Recommendation:

We recommend using OpenZeppelin ReentrancyGuard library - `nonReentrant` modifier.

Alleviation:

The development team heeded our advice and resolved this issue in commit 4a1f0ba1b7302154a7f029d8956df3ae077863e0.

BV-03: Proper usage of "public" and "external" types

Type	Severity	Location
Optimization	Informational	BeefyVaultV4.sol [L89, L165, L180]

Description:

Public functions `getPricePerFullShare` , `proposeStrat` and `upgradeStrat` are never called within the contract, so they could be declared as `external` .

Recommendation:

We recommend changing `public` to `external` for lower execution gas cost.

Alleviation:

The development team heeded our advice and resolved this issue in commit 4a1f0ba1b7302154a7f029d8956df3ae077863e0.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how `block.timestamp` works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a `struct` assignment operation affecting an in-memory `struct` rather than an instorage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of `private` or `delete`.

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a `constructor` assignment imposing different `require` statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as `constant` contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Dead Code

Code that otherwise does not affect the functionality of the codebase and can be safely omitted.

Icons explanation

✓ : Issue resolved

ⓘ : Issue not resolved / Acknowledged. The team will be fixing the issues in their own timeframe.

ⓘ✓ : Issue partially resolved. Not all instances of an issue was resolved.