Absorber Protcol

Security Assessment

March 23th, 2021

For :
Absorber

Disclaimer

CertiK reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security review.

CertiK Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

CertiK Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

CertiK Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

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	Absorber
Description	DeFi
Platform	Binance Smart Chain; Solidity
Codebase	GitHub Repository
Commit	Bscscan Link

Audit Summary

Delivery Date	March. 23th, 2021
Method of Audit	Static Analysis, Manual Review
Consultants Engaged	2
Timeline	Feb. 20th, 2021 - Feb. 22th, 2021, March. 19th, - March. 23th, 2021

Vulnerability Summary

Total Issues	10
Total Critical	0
Total Major	1
Total Minor	0
Total Informational	9

Executive Summary

This report has been prepared for **Absorber.sol** 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.

All of the functions in the protocol have proper access restriction and parameter sanitization where necessary. The equity was found to be calculated correctly for each of the accounts. Most of the findings are optimizational.

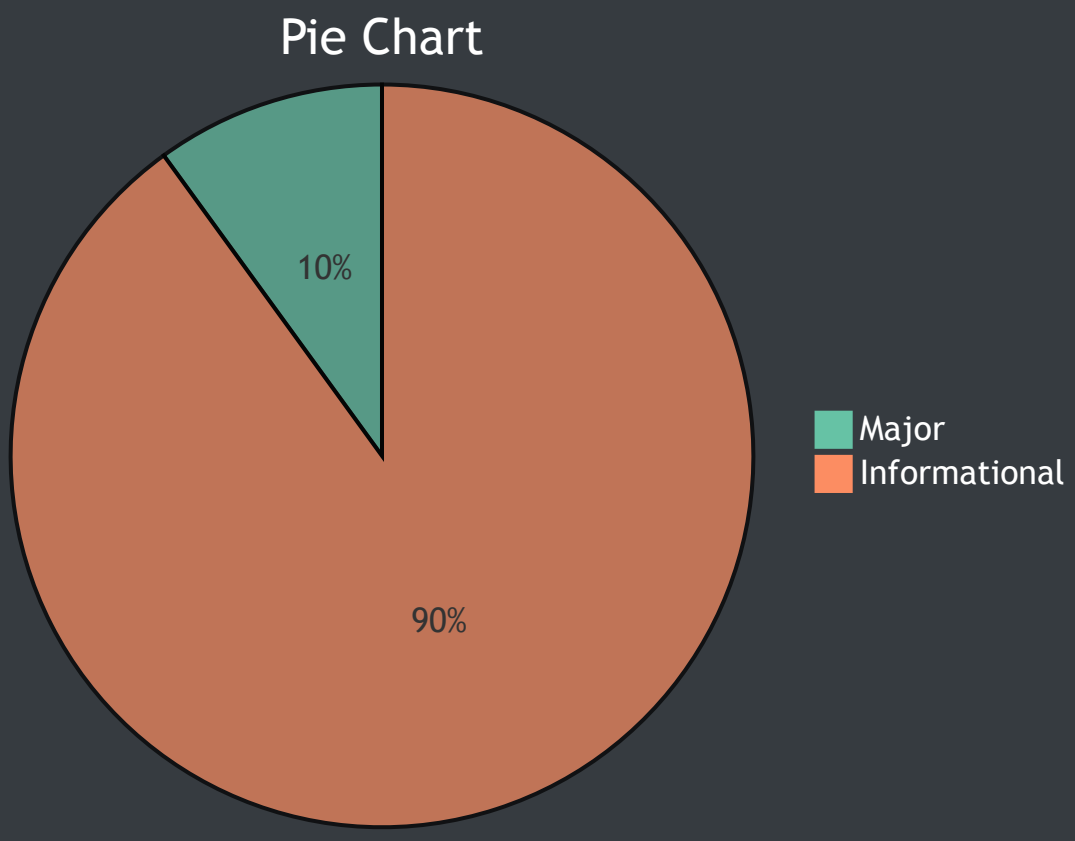
Additionally, to bridge the trust gap between administrator and users, administrator needs to express a sincere attitude with the consideration of the administrator team's anonymousness. The administrator has the responsibility to notify users with the following privileges of the `owner` :

- `owner` can update `feeDivider` , `feeDecimals` and `feePercentage` through `updateFeesAndSwapsEnabled()` function in `absorber.sol` smart contract.
- `owner` can update `feeDecimals` and `feePercentage` through `updateFee()` function in `absorber.sol` smart contract.
- `owner` can update `minTokensBeforeAddToLP` through `updateMinTokensBeforeAddToLP()` function in `absorber.sol` smart contract.
- `owner` can update `swapAndAbsorbEnabled` through `updateSwapAndAbsorbEnabled()` function in `absorber.sol` smart contract.
- `owner` can update `feeDivider` through `changeFeeDivider()` function in `absorber.sol` smart contract.
- `owner` can burn any amount of `_token` assests from any address `_token` through `burnLiq()` function in `absorber.sol` .
- `owner` can exclude and include back any account address through `excludeAccount()` function and `includeAccount()` function in `absorber.sol` .

Client should inform any sensitive changes of project to project's community to improve the trustworthiness of the project. Moreover, any dynamic runtime changes on the protocol should be notified to the community. We also advise client to adopt Multisig, Timelock and/or DAO in the project.

File in Scope

ID	Contract	SHA-256 Checksum
ASB	**Absorber.sol**	ba84ef98b80fcd564588632f0ddb11f90487c8a836b4994aab651fdf67ca9dce


Pie Chart
10%
90%
Major
Informational

ID	Title	Type	Severity	Resolved
ASB-01	Unlocked Compiler Version	Language Specific	Informational	
ASB-02	`_minimumSupply` is never been initialized	Optimization	Informational	
ASB-03	Function Return Value Ignored	Optimization	Informational	
ASB-04	Variable Name Shadowing	Optimization	Informational	
ASB-05	Never Used Variables	Gas Optimization	Informational	
ASB-06	Proper Usage of `public` and `external` type	Gas Optimization	Informational	
ASB-07	Inaccurate Log Message	Gas Optimization	Informational	
ASB-08	Redundant Parameter in Function `burnLiq`	Gas Optimization	Informational	
ASB-09	Merge Redundant If Conditional Branches	Gas Optimization	Informational	
ASB-10	Centralized Risk	Optimization	Major	

ASB-01: Unlocked Compiler Version

Type	Severity	Location
Language Specific	Informational	Absorber.sol

Description:

An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation:

We advise that the compiler versions of codebase is instead locked at the lowest version possible that the full project can be compiled at.

Alleviation:

N/A

ASB-02: `_minimumSupply` is never been initialized

Type	Severity	Location
Optimization	Informational	Absorber.sol L734

Description:

Value of `_minimumSupply` is never been initialized or updated. By default, The value for the any uint is 0.

Recommendation:

We advise client to initialize `_minimumSupply` to a concreate value in order to reflect and inform the minimum supply amount of the project.

Alleviation:

N/A

ASB-03: Function Return Value Ignored

Type	Severity	Location
Optimization	Informational	Absorber.sol L1080, L1160

Description:

Return values of invocation of `addLiquidityETH()` in function `addLiquidity()` and invocation of `transfer()` in function `burnLiq()` are ignored.

Recommendation:

We advise developers to check return values of `transfer()` to check if the transfer is executed without any error.

Alleviation:

N/A

ASB-04: Variable Name Shadowing

Type	Severity	Location
Optimization	Informational	Absorber.sol L843, L986

Description:

The name of parameter `owner` in function `allowance()` and function `_approve()` is shadowing name of function `owner()` in contract `Ownable`

Recommendation:

Rename the parameter name of `owner` in contract `Absorber`

Alleviation:

N/A

ASB-05: Never Used Variables

Type	Severity	Location
Gas Optimization	Informational	Absorber.sol L709

Description:

Variable `transferPaused` and `_balanceOfLpTokens` have never been used throughout the codebase

Recommendation:

Remove variable `transferPaused` and `_balanceOfLpTokens` to save gas consumption.

Alleviation:

N/A

ASB-06: Proper Usage of “public” and “external” type

Type	Severity	Location
Gas Optimization	Informational	Absorber.sol L1246, L1255, L843, L854, L808, L908, L889, L783, L791, L815, L835, L871, L1090

Description:

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays `external` functions are more efficient than "public" functions.

Examples:

- `allocate()` in contract `Absorber`
- `allocationFromToken()` in contract `Absorber`
- `allowance()` in contract `Absorber`
- `approve()` in contract `Absorber`
- `decimals()` in contract `Absorber`
- `decreaseAllowance()` in contract `Absorber`
- `increaseAllowance()` in contract `Absorber`
- `name()` in contract `Absorber`
- `symbol()` in contract `Absorber`
- `totalSupply()` in contract `Absorber`
- `transfer()` in contract `Absorber`
- `transferFrom()` in contract `Absorber`
- `updateFeesAndSwapsEnabled()` in contract `Absorber`

Recommendation:

Consider using the `external` attribute for functions never called from the contract.

Alleviation:

N/A

ASB-07: Inaccurate Log Message

Type	Severity	Location
Gas Optimization	Informational	Absorber.sol L1281

Description:

The message shows in `require(_isExcluded[account], "Account is already excluded");` inaccurately express the correct logging message.

Recommendation:

We advise the client to modify the message into `"Account is not excluded"`

Alleviation:

N/A

ASB-08: Redundant Parameter in Function `burnLiq`

Type	Severity	Location
Gas Optimization	Informational	Absorber.sol L1154

Description:

The parameter `_to` in function `burnLiq()` hasn't been used throughout the logical process.

Recommendation:

We advise the client to refactor the function `burnLiq()` by removing parameter `_to` and `require()` check in L1155 to save gas.

Alleviation:

N/A

ASB-09: Merge Redundant If Conditional Branches

Type	Severity	Location
Gas Optimization	Informational	Absorber.sol L947, L963

Description:

Following if conditional branch at line 946 can be merged into `else` branch in L951

```
...
else if (!_isExcluded[from] && !_isExcluded[to]) {
    _transferStandard(from, address(this), tokensToLock);
...
```

Following if conditional branch at line 962 can be merged into `else` branch in L966

```
...
else if (!_isExcluded[from] && !_isExcluded[to]) {
    _transferStandard(from, to, tokensToTransfer);
...
```

Recommendation:

We advise the client to merge the if conditional branches to save gas

Alleviation:

N/A

ASB-10: Centralized Risk

Type	Severity	Location
Optimization	Major	Absorber.sol

Description:

`owner` is an important role in the contract. The owner address can operate on following functions:

- `updateFeesAndSwapsEnabled()`
- `updateFee()`
- `updateMinTokensBeforeAddToLP()`
- `swapAndAbsorbEnabled`
- `changeFeeDivider()`
- `burnLiq()`
- `excludeAccount()`

Recommendation:

We advise the client to carefully manage project's private key and avoid any potential risks of being hacked. We also advise the client to adopt Multisig, Timelock and/or DAO in the project to manage sensitive role accesses.

Alleviation:

[Absorber Team]: Migration ongoing is open yet until december 2021
Tokens cannot be distributed until then, also in the deployer which i control theres the pool distribution, since theres impossible to mint any more tokens that all we have left, and they are being used for pools at a pretty high rate.

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 the own timeframe.

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