CERTIK

Security Assessment

Wault-Finance

May 4th, 2021

Summary

This report has been prepared for Wault-Finance smart contracts, 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.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Wault-Finance
Description	An standard ERC-20 token smart contract.
Platform	BSC
Language	Solidity
Codebase	1. https://github.com/WaultFinance/WAULT/blob/master/contracts/WAULTx.sol 2. https://bscscan.com/token/0xb64e638e60d154b43f660a6bf8fd8a3b249a6a21
Commits	72f2c6dfed970940dc4b43bd565eb707a13ef6bb

Audit Summary

Delivery Date	May 04, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

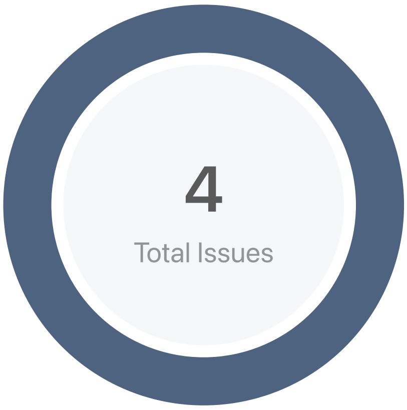
Vulnerability Summary

Total Issues	4
● Critical	0
● Major	0
● Medium	0
● Minor	0
● Informational	4
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
CKP	wault.sol	7d9fc3d2ed192f75af2cb447557983028a4e988ac20f0a3d21b11e40de124d47

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**0** (0.00%)
🟧 **Medium**	**0** (0.00%)
🟨 **Minor**	**0** (0.00%)
🟦 **Informational**	**4** (100.00%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
CKP-01	Inaccurate Solidity version	Language Specific	● Informational	ⓘ Pending
CKP-02	Proper Usage of `public` and `external` type	Gas Optimization	● Informational	ⓘ Pending
CKP-03	Unused internal function	Coding Style	● Informational	ⓘ Pending
CKP-04	Inaccurate result of checking contract address	Logical Issue	● Informational	ⓘ Pending

CKP-01 | Inaccurate Solidity version

Category	Severity	Location	Status
Language Specific	● Informational	wault.sol: 7	ⓘ Pending

Description

Visibility of `constructor` for contract `ERC20` and `Wault` is not specified. This feature is not supported before Solidity 0.7.0. Therefore the Solidity version should not be less than 0.7.0.

Recommendation

We recommend excluding Solidity version 0.6.x.

CKP-02 | Proper Usage of `public` and `external` type

Category	Severity	Location	Status
Gas Optimization	● Informational	wault.sol: 370, 378, 402, 409, 421, 429, 440, 458, 476, 495	ⓘ Pending

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.

Example functions :

- name()
- symbol()
- totalSupply()
- balanceOf(address)
- transfer(address,uint256)
- allowance(address,address)
- approve(address,uint256)
- transferFrom(address,address,uint256)
- increaseAllowance(address,uint256)
- decreaseAllowance(address,uint256)

Recommendation

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

CKP-03 | Unused internal function

Category	Severity	Location	Status
Coding Style	● Informational	wault.sol: 593	ⓘ Pending

Description

Internal function `_setupDecimals()` can never be called externally. Considering it is never called within the contract, it can be removed.

Recommendation

We recommend removing function `_setupDecimals()`.

CKP-04 | Inaccurate result of checking contract address

Category	Severity	Location	Status
Logical Issue	● Informational	wault.sol: 635	ⓘ Pending

Description

`Address.isContract()` does not guarantee `account` is not a contract when it returns `false` (as it is described in the comments from line #623 to line #632).

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings 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 findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings 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 in-storage 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 but rather 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.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is 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 assessment. This report does 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.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.