CertiK Audit Report For Deipool

CertiK Audit Report For

Deipool

Request Date: 2020-09-18

Revision Date: 2020-09-30

Platform Name: EVM

Contents

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 Verification Services Agreement between CertiK and Deipool (the "Company"), or the scope of services/verification, and terms and conditions provided to the Company in connection with the verification (collectively, 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.

About CertiK

CertiK is a technology-led blockchain security company founded by Computer Science professors from Yale University and Columbia University built to prove the security and correctness of smart contracts and blockchain protocols.

CertiK, in partnership with grants from IBM and the Ethereum Foundation, CertiK's mission of every audit is to apply different approaches and detection methods, ranging from manual, static, and dynamic analysis, to ensure that projects are checked against known attacks and potential vulnerabilities. CertiK leverages a team of seasoned engineers and security auditors to apply testing methodologies and assessments to each project, in turn creating a more secure and robust software system.

CertiK has served more than 100 clients with high quality auditing and consulting services, ranging from stablecoins such as Binance's BGBP and Paxos Gold to decentralized oracles

such as Band Protocol and Tellor. CertiK customizes its engineering tool kits, while applying cutting-edge research on smart contracts, for each client on its project to offer a high quality deliverable. For more information: https://certik.io.

Executive Summary

This report has been prepared for **Deipool** to discover issues and vulnerabilities in the source code of their **ERC 20 Token** 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.

Testing Summary

SECURITY LEVEL

File	Vulnerabilities	Result
SavingAccount.sol	1 Major	Pass
SavingAccountParameters.sol	0	Pass
SymbolsLib.sol	0	Pass
TokenInfo.sol	0	Pass
ABDK.sol	0	Pass
Context.sol	0	Pass
IERC20.sol	0	Pass
Ownable.sol	0	Pass
provableAPI.sol	0	Pass
SafeMath.sol	0	Pass
strings.sol	0	Pass
SignedSafeMath.sol	0	Pass

Smart Contract Audit

This report has been prepared as a product of the Smart Contract Audit request by Deipool.

This audit was conducted to discover issues and vulnerabilities in the source code of Deipool's ERC 20 token.

TYPE	Smart Contract
SOURCE CODE	https://github.com/DarionRichie/dip/tree/master/contract
PLATFORM	EVM
LANGUAGE	Solidity
REQUEST DATE	Sep 18, 2020
DELIVERY DATE	Sep 30, 2020
METHODS	A comprehensive examination has been performed using Dynamic Analysis, Static Analysis, and Manual Review.

Review Notes

Introduction

CertiK team was contracted by the **Deipool** team to audit the design and implementation of their ERC 20 token smart contract.

This audit report is only conducted to examine the **Deipool** as 13 files detail information are listed below.

The audited source code link is:

- Source Code:

  contract_last.rar file
- Source Code SHA-256 Checksum

  1) **contract_last.rar** hash

     6d215ee62e773b0ddf0050c9a7be1039c1b20e5da2c3a804a2222a69df784145
  2) **ABDK.sol** hash

     51fb2068a5f1ca5dc299870fd43022b2f484c2f05b646644a9d583361ebdc762
  3) **Context.sol** hash

     b2fc3b3a2d5622dc734e6b527dbc03498cdb042ad12ef9a2b432321e551766cb
  4) **IERC20.sol** hash

     aa9af740081b9ba98d1d3dc26e5f726add6b934e3bc2ee80485dd1286789fae8
  5) **Ownable.sol** hash

     e8bf82364d9b09e2e5352d35eba3a7320212f206a40d7cd202592b2f24b0c9ec
  6) **provableAPI.sol** hash

     cdfa873ba72a833bbcff4451082e62707cfe5b71109bbe5bc9cea50cc07949fc

7) **SafeMath.sol** hash
   04ecf03bf45de096fee2ef714bfd7defce9546670c3534ce8200ddf21ebedff8
8) **SavingAccount.sol** hash
   d762bfe97d413e5b895c55cf71a6a30a5c9da933397d9fee9e1e90d0d9be49f8
9) **SavingAccountParameters.sol** hash
   38fdf955cd927c70b18671c141a2c35cbe4359d177041ed5cbe7bb92708e286b
10) **SignedSafeMath.sol** hash
    6a71b30419f0694b16cf9dd164bb9d5dfec161927ecc45b765b1ed454e4e478a
11) **strings.sol** hash
    771de6034af5ce4167c286f6de4599854ffa4dd00eb197589b77e59a936fda3c
12) **SymbolsLib.sol** hash
    fcf6d24580ebb54735f15ec272eab073dcbc93f6897eb7d4a969317c2abe89aa
13) **TokenInfoLib.sol** hash
    5b5e31b150181b98f9b1ae5426e14dcb0228613463897e0d277f83b98cc0cc37

The goal of this audit was to review the Solidity implementation for its business model, study potential security vulnerabilities, its general design and architecture, and uncover bugs that could compromise the software in production.

The findings of the initial audit have been conveyed to the team behind the contract implementations and the source code is expected to be re-evaluated before another round of auditing has been carried out.

Documentation

The sources of truth regarding the operation of the contracts in scope were lackluster and **are something we advise to be enriched to aid in the legibility of the codebase as well as project**. To help aid our understanding of each contract's functionality we referred to in-line comments and naming conventions.

These were considered the specification, and when discrepancies arose with the actual code behaviour, we consulted with the **Deipool** team or reported an issue.

Summary

The codebase of the project is a typical ERC implementation and the locking mechanism of the token is derived from an officially recognized library, specifically from OpenZeppelin.

Certain optimization steps that we pinpointed in the source code mostly referred to coding standards and inefficiencies, however **1 critical and 2 major vulnerabilities were identified during our audit that solely concerns the specification**.

Certain discrepancies between the expected specification and the implementation of it were identified and were relayed to the team, however they pose no type of vulnerability and concern an optional code path that was unaccounted for.

Recommendations

Overall, the codebase of the contracts should be refactored to assimilate the findings of this report, enforce linters and / or coding styles as well as correct any spelling errors and mistakes that appear throughout the code **to achieve a high standard of code quality and security**.

Findings

Exhibit 1

TITLE	TYPE	SEVERITY	LOCATION
Address of dip token not correct	Optimization	Major	SavingAccount.sol L182

[MAJOR] Description:

The address “0x118b9fabe76e52B4bE24064ecC2630edD1d89067” in the required check is not the address of the dip token.

Recommendations:

Change it to “0xd1517663883e2Acc154178Fb194E80e8bBc29730” which is the address of the dip token.

Alleviation:

Resolved in commit b6cb349eaa0fd7b2581fe007bbeae46f2151d734

File hash 04b152a941145b061b32307f9cc236df32dfb468ef651d346fa7baddc39546d3

Exhibit 2

TITLE	TYPE	SEVERITY	LOCATION
If repay amount more than user loaned	Optimization	Discussion	SavingAccount.sol L210

[DISCUSSION] Description:

If users can repay an amount more than they loaned? If yes, can the surplus obtain interest?

Alleviation:

Deipool Replies: Overpayment will not earn interest.

Exhibit 3

TITLE	TYPE	SEVERITY	LOCATION
If depositAmount contains interest	Optimization	Discussion	SavingAccount.sol L240, 246

[DISCUSSION] Description:

int256 depositedAmount = tokenInfo.addAmount(amount, SUPPLY_APR_PER_SECOND, block.timestamp) - currentBalance;

It seems depositedAmount contains interest since both currentBalance and addAmount contains interest calculation.

Alleviation:

Deipool Replies: This is corrected. Deipool already tested it.

Exhibit 4

TITLE	TYPE	SEVERITY	LOCATION
If liquidate can be called by anyone	Optimization	Discussion	SavingAccount.sol L271

[DISCUSSION] Description:

LIQUIDATE_THREADHOLD is 85, it means if someone calls liquidate, they could get at most 15% of the amount token that targetAddress owned. Does it match Deipool design?

Alleviation:

Deipool Replies: Relevant content will be added in the white paper.

Exhibit 5

TITLE	TYPE	SEVERITY	LOCATION
There are several “+”, ”-”,”*”,”/” operations	Optimization	Informational	SavingAccount.sol L186,L246,L391,L392,L397,L404,L408,L409,L411,L415,L421,L423,L422,L431,L432,L439,L440,L454,L455,L504,L505

[INFORMATIONAL] Description:

It is safer to use operations defined in SafeMath, SignedSafeMath.

Recommendations:

Replace “+” with add, “-” with sub, “*” with mul, “/” with div;

Alleviation:

Resolved in commit b6cb349eaa0fd7b2581fe007bbeae46f2151d734

File hash 04b152a941145b061b32307f9cc236df32dfb468ef651d346fa7baddc39546d3

Exhibit 6

TITLE	TYPE	SEVERITY	LOCATION
Avoid duplicate calculations	Optimization	Minor	SavingAccount.sol L422 & L431

[INFORMATIONAL] Description:

The daily rewards will be added to “totalReward”, and within 180 days, the total DIP amount will be added once more.

Recommendations:

Remove the code in line 422.

Alleviation:

Resolved in commit b6cb349eaa0fd7b2581fe007bbeae46f2151d734

File hash 04b152a941145b061b32307f9cc236df32dfb468ef651d346fa7baddc39546d3

Exhibit 7

TITLE	TYPE	SEVERITY	LOCATION
Simplifying existing code	Optimization	Informational	SavingAccount.sol L402 to L427

[INFORMATIONAL] Description:

The existing code that spans from lines 402 to 427 can be optimized. The “list” array variable is unnecessary. The variable “B” in line 406 is unnecessary.

Recommendations:

Simplifying existing code.

Alleviation:

Partly resolved in commit b6cb349eaa0fd7b2581fe007bbeae46f2151d734. And didn’t remove the “list” variable.

File hash 04b152a941145b061b32307f9cc236df32dfb468ef651d346fa7baddc39546d3

Exhibit 8

TITLE	TYPE	SEVERITY	LOCATION
Lack of the detailed instructions for liquidate	Optimization	Information	White paper 4.4.3

[INFORMATION] Description:

About the liquidation is described briefly in the white paper and doesn't reflect the following two points:

1. Liquidation may be initiated when the asset-liability ratio is greater than 85%.
2. Liquidation will liquidate all the debts and deposits of all types of tokens in the account.

It is recommended to improve the white paper description.

Recommendations:

Could we add more information about the liquidation in the white paper?

Alleviation:

Deipool Replies: Deipool will add these information to the new version white paper.

Exhibit 9

TITLE	TYPE	SEVERITY	LOCATION
Inaccurate require message	Optimization	Informational	SavingAccount.sol L450

[INFORMATIONAL] Description:

The logical judgment expression is 85%, but the required message is 95%.

Recommendations:

The required message should be accurate with the judgment expression.

Alleviation:

Resolved in commit 126e899eff2d22e454d37e3bcd3665b86ac46311

File hash b9bf23fd4fd108b5a83bac35d35af5e672aca3f68fc69c41bd4a4b9ceb979e88

Exhibit 10

TITLE	TYPE	SEVERITY	LOCATION
Method receive and receivemyself looks the same	Optimization	Informational	SavingAccount.sol L467, L484

[INFORMATIONAL] Description:

Method receive and receivemyself looks the same besides function name.

Recommendations:

Remove one function or update one.

Alleviation:

Resolved in commit e5568249bd2b7171f03527a90eaa1179aa6470bb

File hash e3f5b8b74f0dbe52f3f1f399780dc233644621927a65329201675ae106f689fb