CERTIK

Preliminary Comments

ShibaDoge

Jan 12th, 2022

Table of Contents

Summary

This report has been prepared for ShibaDoge to discover issues and vulnerabilities in the source code of the ShibaDoge project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing 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.

Additionally, this audit is based on a premise that all external contracts were implemented safely.

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;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	ShibaDoge
Platform	ethereum
Language	Solidity
Codebase	https://etherscan.io/address/0x6ADb2E268de2aA1aBF6578E4a8119b960E02928F#code
Commit	

Audit Summary

Delivery Date	Jan 12, 2022
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	4	4	0	0	0	0
Medium	0	0	0	0	0	0
Minor	4	4	0	0	0	0
Informational	12	12	0	0	0	0
Discussion	1	1	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
SDS	ShibaDoge.sol	a9c9365e457e11e1a09601ad3e5731b57cbb7dc72136bd9d604a8f90f26ea051

Understandings

Overview

`ShibaDoge` is a deflationary token contract, the token in the contract is `ShibDoge`. The contract uses `_rOwned` and `_tOwned` to record the user's account balance, where `_rOwned` is used to calculate dividends, and `_tOwned` records the user's token balance.

When transfer, if the contract account balance exceeds the set value of `numTokensSellToAddToLiquidity`, the `swapAndLiquify` operation will be executed. First, the token of the contract account balance will be divided according to a certain ratio, one part will be converted into ETH (one part will be used to add liquidity, the other part will be transferred to the `_marketingAddress` address and `_devwallet`), and the other part will be used to add liquidity. The lp will be sent to `address(this)`.

If one of the both parties to the transaction are in the fee exclusion list, then the transaction is free of charge. In addition, there is a special case: transfer from to to is also free of charge. Otherwise, the transaction fee is set according to the specific circumstances of buy or sell(47% liquidity fee and 48% marketing fee). The specific fee can be set in the contract.

Privileged Functions

The contract contains the following privileged functions that are restricted by some modifiers. They are used to modify the contract configurations and address attributes. We grouped these functions below:

The `onlyOwner` modifier:

Contract `Ownable`:

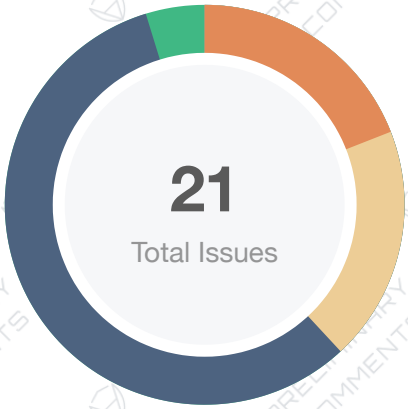
- renounceOwnership()
- transferOwnership(address newOwner)
- lock(uint256 time)

Contract `ShibaDoge`:

- updateMarketingWallet(address payable newAddress)
- updateDevWallet(address payable newAddress)
- updateExchangeWallet(address newAddress)
- updatePartnershipsWallet(address newAddress)
- addBotToBlacklist(address account)
- removeBotFromBlacklist(address account)

- excludeFromReward(address account)
- includeInReward(address account)
- excludeFromFee(address account)
- includeInFee(address account)
- excludeFromLimit(address account)
- includeInLimit(address account)
- setSellFee(uint16 tax, uint16 liquidity, uint16 marketing, uint16 dev, uint16 donation)
- setBuyFee(uint16 tax, uint16 liquidity, uint16 marketing, uint16 dev, uint16 donation)
- setBothFees(uint16 buy_tax, uint16 buy_liquidity, uint16 buy_marketing, uint16 buy_dev, uint16 buy_donation, uint16 sell_tax, uint16 sell_liquidity, uint16 sell_marketing, uint16 sell_dev, uint16 sell_donation)
- setNumTokensSellToAddToLiquidity(uint256 numTokens)
- setMaxTxPercent(uint256 maxTxPercent)
- _setMaxWalletSizePercent(uint256 maxWalletSize)
- setSwapAndLiquifyEnabled(bool _enabled)

Findings



Critical	0 (0.00%)
Major	4 (19.05%)
Medium	0 (0.00%)
Minor	4 (19.05%)
Informational	12 (57.14%)
Discussion	1 (4.76%)

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Risk	**Centralization / Privilege**	**Major**	Pending
GLOBAL-02	Third Party Dependencies	Volatile Code	Minor	Pending
GLOBAL-03	Financial Model	Logical Issue	Minor	Pending
GLOBAL-04	Unlocked Compiler Version	Language Specific	Informational	Pending
GLOBAL-05	Function Visibility Optimization	Gas Optimization	Informational	Pending
GLOBAL-06	Missing Emit Events	Coding Style	Informational	Pending
GLOBAL-07	Inconsistent Integer Types	Coding Style	Informational	Pending
SDS-01	Token Minted To Centralized Address	Logical Issue	Major	Pending
SDS-02	Contract gains non-withdrawable ETH via the `swapAndLiquify` function	Logical Issue	Major	Pending
SDS-03	Centralized Risk In `swapAndLiquify`	**Centralization / Privilege**	**Major**	Pending
SDS-04	Potential Sandwich Attacks	Logical Issue	Minor	Pending
SDS-05	Miscalculation of Max Holding	Mathematical Operations	Minor	Pending
SDS-06	Variables Could Be Declared `Constant` or `Immutable`	Gas Optimization	Informational	Pending

ID	Title	Category	Severity	Status
SDS-07	Unused Event	Coding Style	Informational	Pending
SDS-08	Missing Input Validation	Logical Issue	Informational	Pending
SDS-09	Typos in the contract	Coding Style	Informational	Pending
SDS-10	Error Require Message	Coding Style	Informational	Pending
SDS-11	Hardcode Decimal	Coding Style	Informational	Pending
SDS-12	Redundant code	Logical Issue	Informational	Pending
SDS-13	Return value not handled	Volatile Code	Informational	Pending
SDS-14	The purpose of function `deliver`	Control Flow	Discussion	Pending

GLOBAL-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Pending

Description

In the contract `Ownable`, the role `owner` has the authority over the following function:

- renounceOwnership()
- transferOwnership()
- lock()

In the contract `ShibaDoge`, the role `owner` has the authority over the following function:

- updateMarketingWallet()
- updateDevWallet()
- updateExchangeWallet()
- updatePartnershipsWallet()
- addBotToBlacklist()
- removeBotFromBlacklist()
- excludeFromReward()
- includeInReward()
- excludeFromFee()
- includeInFee()
- excludeFromLimit()
- includeInLimit()
- setSellFee()
- setBuyFee()
- setBothFees()
- setNumTokensSellToAddToLiquidity()
- setMaxTxPercent()
- _setMaxWalletSizePercent()
- setSwapAndLiquifyEnabled()

Additionally, all tokens will be minted to the deployer account.

Any compromise to these accounts may allow the hacker to manipulate the project through these functions.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles. OR

- Remove the risky functionality.

GLOBAL-02 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	Global	Pending

Description

The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

GLOBAL-03 | Financial Model

Category	Severity	Location	Status
Logical Issue	Minor	Global	Pending

Description

When transfer, if one of the both parties to the transaction are in the fee exclusion list, then the transaction is free of charge. In addition, there is a special case: transfer from to to is also free of charge. Otherwise, the transaction fee is set according to the specific circumstances of buy or sell(47% liquidity fee and 48% marketing fee).

If the contract account balance exceeds the set value of `numTokensSellToAddToLiquidity`, the `swapAndLiquify` operation will be executed. First, the token of the contract account balance will be divided according to a certain ratio, one part will be converted into ETH (one part will be used to add liquidity, the other part will be transferred to the `_marketingAddress` address and `_devwallet`), and the other part will be used to add liquidity. The lp will be sent to `address(this)`.

Recommendation

We recommend to publish this feature to the community.

GLOBAL-04 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	Global	Pending

Description

The following contracts have unlocked compiler versions. 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 ambiguity when debugging as compiler specific bugs may occur in the codebase that would be difficult to identify over a span of multiple compiler versions rather than a specific one.

- ShibaDoge.sol

Recommendation

We advise that the compiler version is alternatively locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.6.2` the contract should contain the following line:

```
pragma solidity 0.6.2;
```

GLOBAL-05 | Function Visibility Optimization

Category	Severity	Location	Status
Gas Optimization	Informational	Global	Pending

Description

The following functions are declared as `public` and are not invoked in any of the contracts contained within the project's scope. The functions that are never called internally within the contract should have external visibility.

contract `Ownable`

- `renounceOwnership()` in L510
- `transferOwnership()` in L519
- `lock()` in L533
- `unlock()` in L541

contract `ShibaDoge`

- `transfer()` in L1059
- `approve()` in L1077
- `transferFrom()` in L1086
- `increaseAllowance()` in L1103
- `decreaseAllowance()` in L1116
- `deliver()` in L1144
- `excludeFromReward()` in L1256
- `excludeFromFee()` in L1278
- `includeInFee()` in L1282
- `excludeFromLimit()` in L1286
- `includeInLimit()` in L1290
- `setSwapAndLiquifyEnabled()` in L1363

Recommendation

We advise that the functions' visibility specifiers are set to `external` and the array-based arguments change their data location from `memory` to `calldata`, optimizing the gas cost of the function.

GLOBAL-06 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	Global	Pending

Description

The function that affects the status of sensitive variables should be able to emit events as notifications to customers.

contract `ShibaDoge`

- `deliver()`
- `updateMarketingWallet()`
- `updateDevWallet()`
- `updateExchangeWallet()`
- `updatePartnershipsWallet()`
- `addBotToBlacklist()`
- `removeBotFromBlacklist()`
- `excludeFromReward()`
- `includeInReward()`
- `excludeFromFee()`
- `includeInFee()`
- `excludeFromLimit()`
- `includeInLimit()`
- `setSellFee()`
- `setBuyFee()`
- `setBothFees()`
- `setNumTokensSellToAddToLiquidity()`
- `setMaxTxPercent()`
- `_setMaxWalletSizePercent()`

Recommendation

We advise the client to add events for sensitive actions, and emit them in the function.

GLOBAL-07 | Inconsistent Integer Types

Category	Severity	Location	Status
Coding Style	Informational	Global	Pending

Description

The definition type of `buyFee` and `sellFee` is `uint8`, and contract-related operations use `uint256` to receive calculation results.

Recommendation

We recommend using the uniform int type in contract and using `SafeMath` for math operations.

SDS-01 | Token Minted To Centralized Address

Category	Severity	Location	Status
Logical Issue	Major	ShibaDoge.sol: 996	Pending

Description

The number of tokens that are minted to the centralized address, may raise the community's concerns about the centralization issue.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles. OR
- Remove the risky functionality.

SDS-02 | Contract gains non-withdrawable ETH via the `swapAndLiquify` function

Category	Severity	Location	Status
Logical Issue	Major	ShibaDoge.sol: 1611	Pending

Description

The `swapAndLiquify` function converts half of the `contractTokenBalance` ShibaDoge tokens to ETH. The other half of ShibaDoge tokens and part of the converted ETH are deposited into the ShibaDoge-ETH pool on uniswap as liquidity. For every `swapAndLiquify` function call, a small amount of ETH leftover in the contract. This is because the price of ShibaDoge drops after swapping the first half of ShibaDoge tokens into ETHs, and the other half of ShibaDoge tokens require less than the converted ETH to be paired with it when adding liquidity. The contract doesn't appear to provide a way to withdraw those ETH, and they will be locked in the contract forever.

Recommendation

It's not ideal that more and more ETH are locked into the contract over time. The simplest solution is to add a `withdraw` function in the contract to withdraw ETH. Other approaches that benefit the SafeMoon token holders can be:

- Distribute ETH to ShibaDoge token holders proportional to the amount of token they hold.
- Use leftover ETH to buy back ShibaDoge tokens from the market to increase the price of ShibaDoge.

SDS-03 | Centralized Risk In `swapAndLiquify`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	ShibaDoge.sol: 1611	Pending

Description

In transactions, the linked statements may be called and `ETH` owned by the contract is transferred to the centralized addresses `_marketingAddress` and `_devwallet`. As a result, over time the addresses will accumulate a significant portion of `ETH`. If the addresses are EOAs (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations; AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement. AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles. OR
- Remove the risky functionality.

SDS-04 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Minor	ShibaDoge.sol: 1665~1671, 1679~1686	Pending

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by backrunning (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `swapTokensForEth()`
- `addLiquidity()`

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

SDS-05 | Miscalculation of Max Holding

Category	Severity	Location	Status
Mathematical Operations	Minor	ShibaDoge.sol: 1600	Pending

Description

The transaction may be charged fees, so the max holding of receiver should be `balanceOf(to) + transferAmount`. The fees should not be calculated in the max holding.

Recommendation

We recommend the client to fix this problem by checking max holding without including fees.

SDS-06 | Variables Could Be Declared `Constant` or `Immutable`

Category	Severity	Location	Status
Gas Optimization	Informational	ShibaDoge.sol: 924, 939, 940, 941	Pending

Description

Variables `_tTotal`, `_name`, `_symbol` and `_decimals` could be declared as `constant` since these state variables are never to be changed.

Recommendation

We recommend declaring those variables as `constant`.

SDS-07 | Unused Event

Category	Severity	Location	Status
Coding Style	Informational	ShibaDoge.sol: 978, 979, 981, 983	Pending

Description

The following events are declared but never used:

- `botAddedToBlacklist`
- `botRemovedFromBlacklist`
- `MinTokensBeforeSwapUpdated`
- `SwapAndLiquify`

Recommendation

We recommend removing these events or emitting them in the right places.

SDS-08 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	ShibaDoge.sol: 1216, 1220, 1224, 1228, 1294, 1308, 1322, 1348, 1352, 1611	Pending

Description

The given input is missing the sanity check.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error as below:

1. updateMarketingWallet():

```
require(newAddress != address(0), "newAddress can not be zero address.");
```

2. updateDevWallet():

```
require(newAddress != address(0), "newAddress can not be zero address.");
```

3. updateExchangeWallet():

```
require(newAddress != address(0), "newAddress can not be zero address.");
```

4. updatePartnershipsWallet():

```
require(newAddress != address(0), "newAddress can not be zero address.");
```

5. The initial `liquidity` fee and `marketing` fee are too high, we recommend to set reasonable values for those fees in `setSellFee()`、`setBuyFee()` and `setBothFees()`.

6. setNumTokensSellToAddToLiquidity():

```
require(numTokens < _tTotal, "numTokens must be less than _tTotal.");
```

7. setMaxTxPercent():

```
require(maxTxPercent < 1000, "maxTxPercent must be less than 1000.");
```

8. swapAndLiquify():

```
require(denominator > 0, "denominator must be greater than 0.");
```

SDS-09 | Typos in the contract

Category	Severity	Location	Status
Coding Style	Informational	ShibaDoge.sol: 1368	Pending

Description

`recieve` should be `receive` in the line of comment `//to recieve ETH from uniswapV2Router when swapping`.

Recommendation

We recommend correcting all typos in the contract.

SDS-10 | Error Require Message

Category	Severity	Location	Status
Coding Style	Informational	ShibaDoge.sol: 1558, 1559	Pending

Description

The judgment condition does not match the message.

Recommendation

We advise refactoring the linked codes as below:

```
1558  require(!_isBlackListedBot[from], "from is blacklisted");
1559  require(!_isBlackListedBot[msg.sender], "you are blacklisted");
```

SDS-11 | Hardcode Decimal

Category	Severity	Location	Status
Coding Style	Informational	ShibaDoge.sol: 924, 974, 975, 976	Pending

Description

The constant state variable `_decimals`, at L941, does not be used at the linked statements.

Recommendation

We advise replacing `9` with `_decimals` at the linked statements.

SDS-12 | Redundant code

Category	Severity	Location	Status
Logical Issue	Informational	ShibaDoge.sol: 1710~1712	Pending

Description

The condition `!_isExcluded[sender] && !_isExcluded[recipient]` can be included in `else` .

Recommendation

The following code can be removed:

```
... else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
    _transferStandard(sender, recipient, amount);
} ...
```

SDS-13 | Return value not handled

Category	Severity	Location	Status
Volatile Code	Informational	ShibaDoge.sol: 1679~1686	Pending

Description

The return values of function `addLiquidityETH` are not properly handled.

```
1679 uniswapV2Router.addLiquidityETH{value: ethAmount}(
1680     address(this),
1681     tokenAmount,
1682     0, // slippage is unavoidable
1683     0, // slippage is unavoidable
1684     address(this),
1685     block.timestamp
1686 );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

SDS-14 | The purpose of function `deliver`

Category	Severity	Location	Status
Control Flow	Discussion	ShibaDoge.sol: 1144	Pending

Description

The function `deliver` can be called by anyone. It accepts an uint256 number parameter `tAmount`. The function reduces the token balance of the caller by `rAmount`, which is `tAmount` reduces the transaction fee. Then, the function adds `tAmount` to variable `_tFeeTotal`, which represents the contract's total transaction fee. We wish the team could explain more on the purpose of having such functionality.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

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.

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.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

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 you ("Customer" or 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, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

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.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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.