CERTIK

Security Assessment

Vetter Skylabs Token

Aug 10th, 2022

Table of Contents

Summary

This report has been prepared for Vetter to discover issues and vulnerabilities in the source code of the Vetter Skylabs Token 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.

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	Vetter Skylabs Token
Platform	BSC
Language	Solidity
Codebase	https://github.com/Vetter-ai/vsl-token

Audit Summary

Delivery Date	Aug 10, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

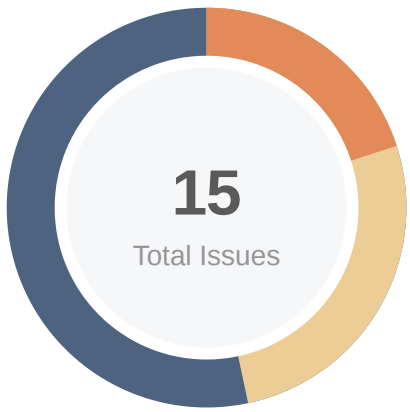
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	3	0	0	0	2	0	1
Medium	0	0	0	0	0	0	0
Minor	4	0	0	4	0	0	0
Informational	8	0	0	2	0	0	6
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
CKP	projects/vsl-token/vslv1.sol	f5f7d4187c8112cf4d83248a4bb1bcf19c21c437bc42962617c70359586b9b5c

Findings



Critical	**0** (0.00%)
Major	**3** (20.00%)
Medium	**0** (0.00%)
Minor	**4** (26.67%)
Informational	**8** (53.33%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CKP-01	Centralization Risks In VSLToken	**Centralization / Privilege**	**Major**	Mitigated
CKP-02	Initial Token Distribution	**Centralization / Privilege**	**Major**	Mitigated
CKP-03	Centralized Risk In `addLiquidity`	**Centralization / Privilege**	**Major**	Resolved
CKP-04	Potential Sandwich Attacks	Logical Issue	Minor	Acknowledged
CKP-05	Third Party Dependencies	Volatile Code	Minor	Acknowledged
CKP-06	Missing Zero Address Validation	Volatile Code	Minor	Acknowledged
CKP-07	Unused Return Value	Volatile Code	Minor	Acknowledged
CKP-11	Missing Emit Events	Coding Style	Informational	Resolved
CKP-12	Missing Error Messages	Coding Style	Informational	Resolved
CKP-13	Function And Variable Naming Doesn'T Match The Operating Environment	Coding Style	Informational	Acknowledged
CKP-14	Unlocked Compiler Version	Language Specific	Informational	Resolved
CKP-15	Declaration Naming Convention	Coding Style	Informational	Acknowledged
CKP-16	Functions With `_` As Name Prefix Are Not `private` Or `internal`	Coding Style	Informational	Resolved

ID	Title	Category	Severity	Status
CKP-17	Visibility Specifiers Missing	Language Specific	Informational	Resolved
CKP-18	Typo In Comments	Coding Style	Informational	Resolved

CKP-01 | Centralization Risks In VSLToken

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/vsl-token/vslv1.sol (vsl-token)	Mitigated

Description

In the contract `VSLToken` the role `_owner` and `_allowedContract` have authority over the following functions:

- `toggleTaxes()`
- `buysellEnabled()`
- `setInitialLaunchTax()`
- `changeRouterVersion()`
- `getExchangeID()`
- `getExchangeAddress()`
- `getExchangeCount()`
- `getAllExchanges()`
- `isExchangeAddress()`
- `setupExchangeAddress()`
- `excludeFromFee()`
- `includeInFee()`
- `getAllAllowedAddresses()`
- `isAllowedContract()`
- `setStakingContract()`
- `getStakingContract()`
- `setExternalAddress()`
- `getExternalAddress()`
- `TransferForeignTokens()`
- `TransferForeignAmount()`
- `TransferBNBToAddress()`
- `TransferAllBNBToAddress()`

Any compromise to the `_owner` and `_allowedContract` accounts may allow the hacker to take advantage of this authority and enable or disable taxes, enable the selling of the token, change the router in use, etc. It

is also worth mentioning that the last four functions would allow the owner and allowed contracts to send funds to arbitrary addresses.

Furthermore, the role `_owner` has authority over the following functions:

- `setupAllowedContract()`
- `transferOwnership()`

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and transfer ownership and add contracts to the `_allowedContract` role.

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., multisignature 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.

Alleviation

[Vetter Skylabs Team]: It is important to note that any contract with settings, like taxation, that can be adjusted will have the "Centralization Risks" warning. The details provided below explain the reason for the functions and how they might effect anyone and access to the contract code to verify the accuracy of the statements. The main risk is loss of access to a wallet that can control the functions listed. This will be mitigated in the future by only providing access to these functions to a DAO or Foundation to make decisions to be executed. At that point, multi-sig and other means can be used to further reduce any risk. In addition, code will show that even if many of these functions are called, there are protections in place. For example, once the ability to buy and sell is turned on it is impossible for any further calls to turn it back off even if access to a wallet was compromised.

See the further details below:

ToggleTaxes - This feature is required to be able to turn the buy-sell tax on or off. Currently, 100% of the buy tax and 33% of the sell tax is allocated to VSL stakers. As noted in the roadmap, the buy-sell tax will be reduced and eventually at zero when enough revenue is generated through the launchpads to reward stakers. We must have the ToogleTaxes feature to fulfill this milestone and eventually turn off buy-sell taxes. (edited)

buysellEnabled - This feature allows for presale participants to stake VSL token before the project launches on PancakeSwap, which increases TVL (total value locked). It's important to note that once live on PancakeSwap this feature can never be reverted back in order to pause or interfere with trading. (edited)

setInitialLaunchTax - This feature provided the option for an anti-bot tax when launched on PancakeSwap. The time window for this function is permanently set for two days in the contract.

changeRouterVersion - If PancakeSwap were to ever change from V2 to V3, there must be a way to move liquidity to the new router. This functions allows for this migration if needed.

Each of the below functions is additional functionality to allow for the possibility to add liquidity to more than one DEX (beyond just PancakeSwap)

getExchangeID, getExchangeAddress, getExchangeCount, getAllExchanges, isExchangeAddress, setupExchangeAddress

excludeFromFee & includeInFee - These functions allow to remove or add a wallet from/to the buy-sell tax. This function is needed to allow for VSL Swap or Vetter Swap to operate without the tax applied.

getAllAllowedAddresses, isAllowedContract - These functions allow the owner to transfer ownership to a DAO in the future before renouncing ownership.

setStakingContract, getStakingContract - These features are required for the option of upgrading the staking contract in the future.

setExternalAddress, getExternalAddress - The feature allows to adjust where development taxes (from the sell tax) are sent. Note that all buy-sell taxes will be set to zero in the future.

TransferForeignTokens, TransferForeignAmount, TransferBNBToAddress, TransferAllBNBToAddress - These features allow for the ability to remove tokens and/or BNB that had been sent to the token contract address, which no one should ever be doing.

CKP-02 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/vsl-token/vslv1.sol (vsl-token): 265~274	Mitigated

Description

All of the `VSL` tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute `VSL` tokens without obtaining the consensus of the community.

Recommendation

We recommend the team to be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

The distribution process occurs before release on PancakeSwap. Once VSL is released on PancakeSwap, tokens will have already been allocated to their appropriate locations as outlined in the whitepaper under Skylabs Presale, tokenomics.

CKP-03 | Centralized Risk In `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/vsl-token/vslv1.sol (vsl-token): 377	Resolved

Description

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `VSLToken-BNB` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens. If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract based accounts with enhanced security practices, f.e. Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

LP will no longer go to owner and will instead go to the contract itself (to be able to be handled by the DAO/Foundation in the future) to add the LP to an internal Vault Contract.

CKP-04 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Minor	projects/vsl-token/vslv1.sol (vsl-token): 377, 406	Acknowledged

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 `swapExactTokensForETHSupportingFeeOnTransferTokens()` and `addLiquidityETH()` functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks.

Recommendation

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

CKP-05 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	projects/vsl-token/vslv1.sol (vsl-token): 215	Acknowledged

Description

The contract is serving as the underlying entity to interact with PancakeSwap. The scope of the audit treats 3rd party entities as black boxes and assume 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 understand that the business logic of `VSLToken` requires interaction with PancakeSwap. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

CKP-06 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	projects/vsl-token/vslv1.sol (vsl-token): 799, 811, 851, 858	Acknowledged

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

Recommendation

We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

CKP-07 | Unused Return Value

Category	Severity	Location	Status
Volatile Code	Minor	projects/vsl-token/vslv1.sol (vsl-token): 377	Acknowledged

Description

The return value of an external call is not stored in a local or state variable.

Recommendation

We recommend checking or using the return values of all external function calls.

CKP-11 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	projects/vsl-token/vslv1.sol (vsl-token): 161, 645, 709, 733, 739, 745, 797, 809	Resolved

Description

There should always be events emitted in the sensitive functions that are controlled by centralization roles.

Recommendation

It is recommended emitting events for the sensitive functions that are controlled by centralization roles.

Alleviation

Most of the functions that update the state are now emitting events.

CKP-12 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	projects/vsl-token/vslv1.sol (vsl-token): 170	Resolved

Description

The **require** can be used to check for conditions and throw an exception if the condition is not met. It is better to provide a string message containing details about the error that will be passed back to the caller.

Recommendation

We advise adding error messages to the linked **require** statements.

Alleviation

The `require` statement is now sending an error message.

CKP-13 | Function And Variable Naming Doesn'T Match The Operating Environment

Category	Severity	Location	Status
Coding Style	Informational	projects/vsl-token/vslv1.sol (vsl-token)	Acknowledged

Description

The VSLToken contract mentions Ethereum names (such as ETH, UniSwap), as well as Binance Smart Chain names (such as BNB, UniSwap, PinkSwap).

Recommendation

Update the naming to the right environment in use for coherence and readability purposes.

CKP-14 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	projects/vsl-token/vslv1.sol (vsl-token): 3	Resolved

Description

The contract has unlocked compiler version. 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 version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.4` the contract should contain the following line:

```
pragma solidity 0.8.4;
```

Alleviation

The pragma version has been set to `0.8.14`.

CKP-15 | Declaration Naming Convention

Category	Severity	Location	Status
Coding Style	Informational	projects/vsl-token/vslv1.sol (vsl-token): 67, 68, 69, 198, 199, 201, 203, 215, 225, 233, 239, 243, 745, 823, 834, 848, 855	Acknowledged

Description

One or more declarations do not conform to the Solidity style guide with regards to its naming convention.

Particularly:

- `camelCase`: Should be applied to function names, argument names, local and state variable names, modifiers
- `UPPER_CASE`: Should be applied to `constant` variables
- `CapWords`: Should be applied to contract names, struct names, event names and enums

```
198     string private constant _name = "V V V Token";
```

- Constant variable `_name` is not in `UPPER_CASE`.

```
199     string private constant _symbol = "VVV";
```

- Constant variable `_symbol` is not in `UPPER_CASE`.

```
201     uint8  private constant _decimals = 9;
```

- Constant variable `_decimals` is not in `UPPER_CASE`.

```
203     uint256 private constant _totalTokens = 25 * 10**9 * 10**_decimals;     // 25 Billion Tokens
```

- Constant variable `_totalTokens` is not in `UPPER_CASE`.

```
215     address payable constant private addressV2Router = payable(0xBBe737384C2A26B15E23a181BDfBd9Ec49E00248);
```

- Constant variable `addressV2Router` is not in `UPPER_CASE`.

```
225     uint256 private constant _maxHighTaxTime = 2 days;  // Maximum time the tax can
be set to a higher rate before cap sets in
```

- Constant variable `_maxHighTaxTime` is not in `UPPER_CASE`.

```
233     uint256 private constant _marketTaxCap = 100;       // 10% max
```

- Constant variable `_marketTaxCap` is not in `UPPER_CASE`.

```
239     uint256 private constant _liquidityCap = 20;        // 2% max
```

- Constant variable `_liquidityCap` is not in `UPPER_CASE`.

```
243     uint256 private constant _royaltyCap = 150;         // 15% max
```

- Constant variable `_royaltyCap` is not in `UPPER_CASE`.

```
245     uint256 private constant _initialCap = 800;         // 80% max
```

- Constant variable `_initialCap` is not in `UPPER_CASE`.

```
67    function DOMAIN_SEPARATOR() external view returns (bytes32);
```

- Function `DOMAIN_SEPARATOR` is not in `camelCase`.

```
68    function PERMIT_TYPEHASH() external pure returns (bytes32);
```

- Function `PERMIT_TYPEHASH` is not in `camelCase`.

```
69    function MINIMUM_LIQUIDITY() external pure returns (uint256);
```

- Function `MINIMUM_LIQUIDITY` is not in `camelCase`.

```
823     function TransferForeignTokens(address _token, address _to) external
onlyAllowedContract returns (bool _sent)
```

- Function `TransferForeignTokens` is not in `camelCase`.

```
834     function TransferForeignAmount(address _token, address _to, uint256 _maxAmount)
external onlyAllowedContract returns (bool _sent)
```

- Function `TransferForeignAmount` is not in `camelCase`.

```
848     function TransferBNBToAddress(address payable recipient, uint256 amount)
external onlyAllowedContract
```

- Function `TransferBNBToAddress` is not in `camelCase`.

```
855     function TransferAllBNBToAddress(address payable recipient) external
onlyAllowedContract
```

- Function `TransferAllBNBToAddress` is not in `camelCase`.

Recommendation

We recommend adjusting those variable and function names to properly conform to Solidity's naming convention.

CKP-16 | Functions With `_` As Name Prefix Are Not `private` Or `internal`

Category	Severity	Location	Status
Coding Style	Informational	projects/vsl-token/vslv1.sol (vsl-token): 761	Resolved

Description

Functions with names starting with `_` should be declared as `private`/`internal`.

Recommendation

Consider changing function visibility to private or removing `_` from the start of the function name.

Alleviation

The name of the mentioned function has been updated so it does not start with an underscore (`_`). This way, it is not confused by an internal function.

However, it is worth mentioning that it is now not following the `camelCase` naming standard.

CKP-17 | Visibility Specifiers Missing

Category	Severity	Location	Status
Language Specific	Informational	projects/vsl-token/vslv1.sol (vsl-token): 253	Resolved

Description

The linked variable declarations do not have a visibility specifier explicitly set.

Recommendation

Inconsistencies in the default visibility the Solidity compilers impose can cause issues in the functionality of the codebase. We advise that visibility specifiers for the linked variables are explicitly set.

Alleviation

The visibility has been set to `private`.

CKP-18 | Typo In Comments

Category	Severity	Location	Status
Coding Style	Informational	projects/vsl-token/vslv1.sol (vsl-token): 236, 808, 814	Resolved

Description

There are some typos in the highlighted comments.

Recommendation

It is recommended to fix the typos for readability purposes.

Alleviation

The typos have been fixed.

Optimizations

ID	Title	Category	Severity	Status
CKP-08	Function Should Be Declared External	Gas Optimization	Optimization	Resolved
CKP-09	Unused State Variable	Gas Optimization	Optimization	Resolved
CKP-10	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	Resolved

CKP-08 | Function Should Be Declared External

Category	Severity	Location	Status
Gas Optimization	Optimization	projects/vsl-token/vslv1.sol (vsl-token): 161, 168, 301, 320, 560, 587, 598, 604, 610, 624, 678, 684	Resolved

Description

The functions which are never called internally within the contract should have external visibility for gas optimization.

Recommendation

We advise to change the visibility of the aforementioned functions to `external`.

CKP-09 | Unused State Variable

Category	Severity	Location	Status
Gas Optimization	Optimization	projects/vsl-token/vslv1.sol (vsl-token): 215	Resolved

Description

The variable `addressV2Router` in `VSLToken` is never used in `VSLToken`.

Recommendation

We advise removing the unused variables.

Alleviation

The unused state variable `addressV2Router` has been commented out.

CKP-10 | Variables That Could Be Declared As Immutable

Category	Severity	Location	Status
Gas Optimization	Optimization	projects/vsl-token/vslv1.sol (vsl-token): 224	Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

The `_timeTokenLaunched` state variable is now declared as `immutable`.

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.

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.

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.