CERTIK

Code Security Assessment

WOOFi II

Feb 7th, 2022

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- BSW-01 : Missing emit events
- BSW-02 : Lack of Input Validation
- BSW-03 : Fee Collectors
- SCW-01 : Third Party Dependencies
- SLP-01 : Third Party Dependencies
- SLP-02 : Lack of Input Validation
- SLP-03 : Missing Error Messages
- SLP-04 : Return value not handled
- SLP-05 : Wrong Path To Swap `lpToken1`
- VWO-01 : Address Type Could Be Indexed In Events
- VWO-02 : Magic Numbers
- VWO-03 : Contract Locks Ether
- VWO-04 : Lack Of Stop Strategy In `setupStrat`
- WOO-01 : Centralization Related Risks
- WOO-02 : Unknown Imported Source Files
- WOO-03 : Strategy not Support `WBNB`

Appendix

Disclaimer

About

Summary

This report has been prepared for WOOFi II to discover issues and vulnerabilities in the source code of the WOOFi II 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	WOOFi II
Platform	bsc
Language	Solidity
Codebase	woofi_earn_Jan_27.zip, woofi_earn_Jan_30.zip, Vault.sol
Commit	Shasum256 of the zip file: 29ed6af071345ad457e840ff829e5f32401756e8a3a814f3728f956da5fe10bf Shasum256 of the zip file: 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434 Shasum256 of Vault.sol: 3e0b032d592ab7747921503995f6b2ff31afc9d11075ee00817b81ca8486eef6

Audit Summary

Delivery Date	Feb 07, 2022
Audit Methodology	Static Analysis, Manual Review

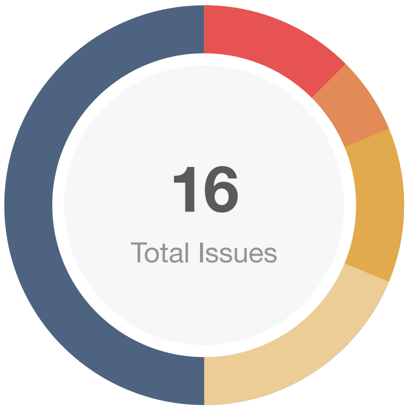
Vulnerability Summary

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

Audit Scope

ID	File	SHA256 Checksum
BSW	BaseStrategy.sol	97555669c004bce4a97ee6ac946120dc627f3b2f55361918020fd7c8da310b03
SCW	StrategyCake.sol	ae4360441f5a939cccd36cbe9310ff992b6c453d5268ae53225ef87d5490480b
SLP	StrategyLP.sol	6e32b8fd74300c2e09a2e02867e3186860d8709956e6f45b52c904ad57d453e8
VWO	Vault.sol	6e5d6f1ba87d13951fecbfe4bff32a30178e94b8ab6ea11ceec2527d89b00c04

Findings



🟥	**Critical**	**2** (12.50%)
🟧	**Major**	**1** (6.25%)
🟨	**Medium**	**2** (12.50%)
🟨	**Minor**	**3** (18.75%)
🟦	**Informational**	**8** (50.00%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
BSW-01	Missing emit events	Coding Style	● Informational	Partially Resolved
BSW-02	Lack of Input Validation	Volatile Code	● Informational	✓ Resolved
BSW-03	Fee Collectors	**Centralization / Privilege**	● **Medium**	ⓘ Acknowledged
SCW-01	Third Party Dependencies	Volatile Code	● Minor	ⓘ Acknowledged
SLP-01	Third Party Dependencies	Volatile Code	● Minor	ⓘ Acknowledged
SLP-02	Lack of Input Validation	Volatile Code	● Informational	✓ Resolved
SLP-03	Missing Error Messages	Coding Style	● Informational	✓ Resolved
SLP-04	Return value not handled	Volatile Code	● Informational	ⓘ Acknowledged
SLP-05	Wrong Path To Swap `lpToken1`	Logical Issue	● Critical	✓ Resolved
VWO-01	Address Type Could Be Indexed In Events	Language Specific	● Informational	✓ Resolved
VWO-02	Magic Numbers	Coding Style	● Informational	ⓘ Acknowledged
VWO-03	Contract Locks Ether	Logical Issue	● Medium	✓ Resolved
VWO-04	Lack Of Stop Strategy In `setupStrat`	Logical Issue	● Minor	✓ Resolved

ID	Title	Category	Severity	Status
WOO-01	Centralization Related Risks	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
WOO-02	Unknown Imported Source Files	Logical Issue	● Informational	ⓘ Acknowledged
WOO-03	Strategy not Support `WBNB`	Logical Issue	● Critical	✓ Resolved

BSW-01 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	BaseStrategy.sol (v2): 104~124	Partially 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

The development team partially resolved this issue in zip source files which shasum is 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

BSW-02 | Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	BaseStrategy.sol (v2): 43	✓ Resolved

Description

The given input `initAccessManager` is missing the sanity checks for ensuring non-zero value.

Recommendation

We advise the client to add the following input validation:

```
require(initAccessManager != address(0), "initAccessManager is a zero address");
```

Alleviation

The development team resolved this issue in zip source files which shasum is 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

BSW-03 | Fee Collectors

Category	Severity	Location	Status
Centralization / Privilege	● **Medium**	BaseStrategy.sol (v2): 35~36	ⓘ Acknowledged

Description

There are some fee collectors, i.e. `performanceTreasury` and `withdrawalTreasury`, over time, these accounts would gain more and more fees.

Recommendation

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

No Alleviation.

SCW-01 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	StrategyCake.sol (v2): 19	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `Pancake MasterChef` protocols. 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 `StrategyCake` requires interaction with `Pancake MasterChef`. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Woofi]: We will keep close eye on Pancake Masterchef. And we only deal with the top-tier dex or lending platforms on BSC.

SLP-01 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	● Minor	StrategyLP.sol (v2): 34~35	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party `Pancake MasterChef` and `PancakeSwap` protocols. 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 `StrategyLP` requires interaction with `Pancake MasterChef` and `PancakeSwap`. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Woofi]: We will keep close eye on Pancake Masterchef. And we only deal with the top-tier dex or lending platforms on BSC.

SLP-02 | Lack Of Input Validation

Category	Severity	Location	Status
Volatile Code	● Informational	StrategyLP.sol (v2): 45	✓ Resolved

Description

The given input `initWant` is missing the sanity checks for ensuring non-zero value.

Recommendation

We advise the client to add the following input validation:

```
require(initWant != address(0), "initWant is a zero address");
```

Alleviation

The development team resolved this issue in zip source files which shasum is 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

SLP-03 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	● Informational	StrategyLP.sol (v2): 63~64	✓ 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 development team resolved this issue in zip source files which shasum is 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

SLP-04 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	● Informational	StrategyLP.sol (v2): 155	ⓘ Acknowledged

Description

The return value of function `addLiquidity` is not properly handled.

```
IPancakeRouter(uniRouter).addLiquidity(lpToken0, lpToken1, lp0Balance, lp1Balance, 0, 0,
address(this), now);
```

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.

Alleviation

[Woofi]:double-checked the pancake router interface and https://docs.uniswap.org/protocol/V2/reference/smart-contracts/router-01 , we don't need to check the return value here. If anything wrong, the tx just got reverted; or there's no LP for us to deposit for farming.

SLP-05 | Wrong Path To Swap `lpToken1`

Category	Severity	Location	Status
Logical Issue	● Critical	StrategyLP.sol (v2): 150	✓ Resolved

Description

```
if (lpToken1 != reward) {
    IPancakeRouter(uniRouter).swapExactTokensForTokens(rewardHalf, 0, rewardToLP0Route,
address(this), now);
}
```

The above code wants to swap reward to `lpToken1`, but it uses `rewardToLP0Route` as the swap path.

Recommendation

Consider changing the values of the `path` to `rewardToLP1Route`:

```
if (lpToken1 != reward) {
    IPancakeRouter(uniRouter).swapExactTokensForTokens(rewardHalf, 0, rewardToLP1Route,
address(this), now);
}
```

Alleviation

The development team resolved this issue in zip source files which shasum is 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

VWO-01 | Address Type Could Be Indexed In Events

Category	Severity	Location	Status
Language Specific	● Informational	Vault.sol (v2): 35~36	Resolved

Description

It is recommended to add `indexed` keyword for parameters in events, which makes it easier for users to navigate event logs.

Recommendation

We advise the client to add keyword `indexed` in the declaration of events.

```
event NewStratCandidate(address indexed implementation);
event UpgradeStrat(address indexed implementation);
```

Alleviation

The development team resolved this issue in zip source files which shasum is 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

VWO-02 | Magic Numbers

Category	Severity	Location	Status
Coding Style	● Informational	Vault.sol (v2): 178	ⓘ Acknowledged

Description

The linked magic numbers should be set as `constant` and `internal` contract-level variables with a self-explanatory variable name as well as accompanying comments when necessary. This type of declaration is functionally equivalent to the current implementation as `constant` variables that are `internal` or `private` are simply replaced in the codebase with their literal value.

```
stratCandidate.proposedTime = 5000000000;
```

Recommendation

We advise the team adds proper documentation specifying the purpose of the linked numbers.

Alleviation

[Woofi]:This number set is from the code from BeefyFinance, not defining the const value to save the gas.

VWO-03 | Contract Locks Ether

Category	Severity	Location	Status
Logical Issue	● Medium	Vault.sol (v2): 193	✓ Resolved

Description

The contract has payable fallback functions but without a withdrawal capacity. Hence every Ether sent to the contract will be lost.

Recommendation

We advise rewriting the functions to avoid loss of Ether or adding an ether withdrawal feature. An example:

```
    function withdraw(address account) external onlyAdmin {
        payable(account).transfer(address(this).balance);
    }
```

Alleviation

The development team resolved this issue in a single source file which shasum is 3e0b032d592ab7747921503995f6b2ff31afc9d11075ee00817b81ca8486eef6.

VWO-04 | Lack Of Stop Strategy In `setupStrat`

Category	Severity	Location	Status
Logical Issue	● Minor	Vault.sol (v2): 156~160	✓ Resolved

Description

In `setupStrat` function, the old strategy should be stopped before it is updated. Otherwise, all the tokens deposited in the old strategy will not be returned back to the vault contract.

Recommendation

Consider stopping the old strategy before it is updated and depositing all tokens to the new strategy:

```
function setupStrat(address _strat) public onlyAdmin {
    require(_strat != address(0), 'Vault: STRAT_ALREADY_SET');
    require(address(this) == IStrategy(_strat).vault(), 'Vault: STRAT_VAULT_INVALID');
    strategy.retireStrat();
    strategy = IStrategy(_strat);
    earn();
}
```

Alleviation

The development team resolved this issue in zip source files which shasum is 1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

WOO-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	● Major	BaseStrategy.sol (v2): 104~136, 53~58 StrategyCake.sol (v2): 28~57, 81~96 StrategyLP.sol (v2): 74~105, 158~173 Vault.sol (v2): 156~191	ⓘ Acknowledged

Description

In the contract `BaseStrategy`, the role `_owner` and `vaultAdmin` have authority over the following functions:

- `setPerformanceFee(uint256 fee)`
- `setWithdrawalFee(uint256 fee)`
- `setPerformanceTreasury(address treasury)`
- `setWithdrawalTreasury(address treasury)`
- `setHarvestOnDeposit(bool newHarvestOnDeposit)`
- `pause()`
- `unpause()`

Any compromise to the `_owner` or `vaultAdmin` account may allow a hacker to take advantage of this authority.

In the contract `BaseStrategy`, the role `vault` has authority over the following functions:

- `beforeDeposit()`

Any compromise to the `vault` account may allow a hacker to take advantage of this authority.

In the contract `StrategyCake`, the role `_owner` and `vaultAdmin` have authority over the following functions:

- `emergencyExit()`

Any compromise to the `_owner` or `vaultAdmin` account may allow a hacker to take advantage of this authority.

In the contract `StrategyCake`, the role `vault` has authority over the following functions:

- `retireStrat()`

- `withdraw(uint256 amount)`
- `harvest()`

Any compromise to the `vault` account may allow a hacker to take advantage of this authority.

In the contract `StrategyLP`, the role `_owner` and `vaultAdmin` have authority over the following functions:

- `emergencyExit()`

Any compromise to the `_owner` or `vaultAdmin` account may allow a hacker to take advantage of this authority.

In the contract `StrategyLP`, the role `vault` has authority over the following functions:

- `retireStrat()`
- `withdraw(uint256 amount)`
- `harvest()`

Any compromise to the `vault` account may allow a hacker to take advantage of this authority.

In the contract `Vault`, the role `_owner` and `vaultAdmin` have authority over the following functions:

- `setupStrat(address _strat)`
- `proposeStrat(address _implementation)`
- `upgradeStrat()`
- `inCaseTokensGetStuck(address stuckToken)`

Any compromise to the `_owner` or `vaultAdmin` account may allow a hacker to take advantage of this authority.

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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

[Woofi]: The owner and access manager will be a 3/5 multi-sig wallet by Gnosis Safe.

WOO-02 | Unknown Imported Source Files

Category	Severity	Location	Status
Logical Issue	● Informational	Vault.sol (v2): 4~14 StrategyLP.sol (v2): 4~16 StrategyCake.sol (v2): 4~9 BaseStrategy.sol (v2): 4~13	ⓘ Acknowledged

Description

The aforementioned imported files are unknown.

Recommendation

These unknown imported files are out of audit scope.

Alleviation

[WOOFi Team]: Openzepplin imports.

WOO-03 | Strategy Not Support `WBNB`

Category	Severity	Location	Status
Logical Issue	● Critical	Vault.sol (v2): 156~160, 169~173, 85~86 StrategyLP.sol (v2): 45~69, 33 StrategyCake.sol (v2): 22	✓ Resolved

Description

The strategy contract `StrategyCake` only supports the `Cake` token, the strategy contract `StrategyLP` supports the `Cake` token and lp-style tokens. Since both of the two contracts do not support `WBNB`, if the principal token `want` in `Vault` is `WBNB` and any strategy is activated, all the `WBNB` tokens will be deposited into the strategy contract and lost forever.

This incident will happen in any other token which is not `Cake` token or `lp` token which is not in `Pancake` Farm.

Recommendation

We advise the client to consider the following workarounds.

- **Case 1**: The `want` is `WBNB`.

Deactivate the strategy. All the WBNBs will be stored in the `Vault` contract account.

- **Case 2**: The `want` is any other token.

Check whether the `want` token of `Vault` equals the `want` token of the strategy contract or not in the function `setupStrat` and `upgradeStrat`. An example.

```
    function setupStrat(address _strat) public onlyAdmin {
        require(_strat != address(0), 'Vault: STRAT_ALREADY_SET');
        require(address(this) == IStrategy(_strat).vault(), 'Vault:
STRAT_VAULT_INVALID');
        require(address(want) == IStrategy(_strat).want(), 'Vault: TOKEN_VAULT_INVALID');
        strategy = IStrategy(_strat);
    }
    //...
    function upgradeStrat() public onlyAdmin {
        require(stratCandidate.implementation != address(0), 'Vault: NO_CANDIDATE');
        require(stratCandidate.proposedTime.add(48 hours) < block.timestamp, 'Vault:
TIME_INVALID');
```

```
        require(address(want) == IStrategy(stratCandidate.implementation).want(), 'Vault:
TOKEN_VAULT_INVALID');
        //....
    }
```

Alleviation

The development team resolved this issue in zip source files which shasum is
1eea647cbb41b0a63827befda410ec1ed8b36d83ebad9b99f0eb27a5bc73b434.

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.

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

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.