Security Assessment

WOOFi III

Jul 5th, 2022

Table of Contents

Summary

Overview

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

Understanding

- External Dependencies
- Privileged Roles

Findings

- EAR-01 : Centralization Related Risks
- EAR-02 : Third Party Dependencies
- EAR-03 : Missing Input Validation
- EAR-04 : Unknown Imported Source File
- EAR-05 : Missing Error Messages
- EAR-06 : Missing Emit Events
- WLM-01 : Divide Before Multiply
- WSC-01 : Fee Collectors
- WSC-02 : Unused `Pausable` Feature
- WSC-03 : Incompatibility With Deflationary Tokens

Appendix

Disclaimer

About

Summary

It is noted that the lender role has the ability to transfer assets in the protocol without any collateral besides updating the interest rate. Any comprise of the lender account will allow the hacker to exhaust all the assets of the protocol.

Overview

Project Summary

Project Name	WOOFi III
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/woonetwork/woofi_swap_smart_contracts/tree/main/contracts/earn
Commit	133080266adc4fcd4ca0450b99539e81cb59308b 7bbb8a6971e785102ebaf5d308fe812711025baa (final report)

Audit Summary

Delivery Date	Jul 05, 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	1	0	0	1	0	0	0
Medium	1	0	0	1	0	0	0
Minor	4	0	0	3	0	0	1
Informational	4	0	0	2	0	1	1
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	Repo	Commit	File	SHA256 Checksum
WLM	woonetwork/woofi_swap_smart_contracts	1330802	WooLendingManager.sol	0216279c6cae001f5bc1eae5ee4ba93c6ef85432d12578ca261ccb1e4899d46d
WSC	woonetwork/woofi_swap_smart_contracts	1330802	WooSuperChargerVault.sol	8d38c1248f9888eca99504be016585719a29698aa5a21de4c2ab5ff24d0c1de7
WWM	woonetwork/woofi_swap_smart_contracts	1330802	WooWithdrawManager.sol	650a0f417b08e5757978bc5f26d2dde14b7b31853099325e2deb125d9b6397a6

Understanding

External Dependencies

The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third parties can be compromised and this may lead to lost or stolen assets.

There are a few depending injection contracts or addresses in the current project:

- `weth`, `want`, `accessManager`, `superChargerVault` for contract `WooLendingManager`;
- `weth`, `want`, `accessManager`, `reserveVault`, `lendingManager`, `withdrawManager` for contract `WooSuperChargerVault`;
- `weth`, `want`, `accessManager`, `superChargerVault` for contract `WooWithdrawManager`;

In addition, the contract is serving as the underlying entity to interact with third-party contracts and interfaces:

- `IERC20`, `IWooAccessManager`, `Ownable`, `ReentrancyGuard`, `SafeERC20`, `SafeMath`, `TransferHelper` for contract `WooLendingManager`;
- `ERC20`, `EnumerableSet`, `IERC20`, `IVaultV2`, `IWETH`, `IWooAccessManager`, `Ownable`, `Pausable`, `ReentrancyGuard`, `SafeERC20`, `SafeMath`, `TransferHelper` for contract `WooSuperChargerVault`;
- `IERC20`, `IWETH`, `IWooAccessManager`, `Ownable`, `ReentrancyGuard`, `SafeERC20`, `SafeMath`, `TransferHelper` for contract `WooWithdrawManager`

We assume these vulnerable actors and implement proper logic to collaborate with the current project.

Privileged Roles

The following roles are adopted to enforce the access control:

In the contract `WooLendingManager`

- Role `_owner` is adopted to update configurations of the contract and transfer assets.
- Role `isLender` is adopted to borrow or repay assets and update configurations of the contract.
- Role `superChargerVault` is adopted to update configurations of the contract.

In the contract `WooSuperChargerVault`

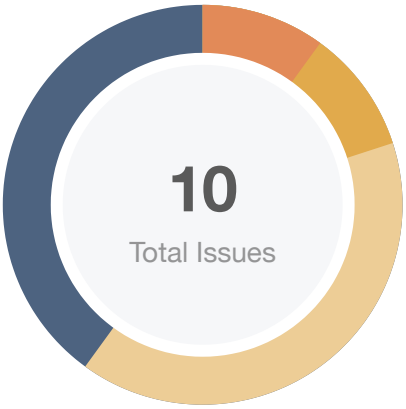
- Role `_owner` is adopted to update configurations of the contract and transfer assets.
- Role `lendingManager` is adopted to borrow or repay assets.
- Role `onlyAdmin` is adopted to start/end the weekly settlement.

In the contract `WooWithdrawManager`

- Role `_owner` is adopted to update configurations of the contract and transfer assets.
- Role `superChargerVault` is adopted to change the withdrawable amount of the contract.
- Role `onlyAdmin` is adopted to update configurations of the contract.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned related functions should be also considered to move to the execution queue of `Timelock` contract.

Findings



Critical	**0** (0.00%)
Major	**1** (10.00%)
Medium	**1** (10.00%)
Minor	**4** (40.00%)
Informational	**4** (40.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
EAR-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
EAR-02	Third Party Dependencies	Volatile Code	Minor	Acknowledged
EAR-03	Missing Input Validation	Volatile Code	Minor	Acknowledged
EAR-04	Unknown Imported Source File	Logical Issue	Informational	Acknowledged
EAR-05	Missing Error Messages	Coding Style	Informational	Acknowledged
EAR-06	Missing Emit Events	Coding Style	Informational	Partially Resolved
WLM-01	Divide Before Multiply	Mathematical Operations	Informational	Resolved
WSC-01	Fee Collectors	**Centralization / Privilege**	**Medium**	Acknowledged
WSC-02	Unused `Pausable` Feature	Logical Issue, Volatile Code	Minor	Resolved
WSC-03	Incompatibility With Deflationary Tokens	Volatile Code	Minor	Acknowledged

EAR-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	WooLendingManager.sol: 73, 104, 108, 148, 153, 165, 169, 174, 181, 206; WooSuperChargerVault.sol: 112, 243, 253, 277, 286, 320, 330, 334; WooWithdrawManager.sol: 72, 97, 101, 120	Acknowledged

Description

In the contract `WooLendingManager`, the role `_owner` has authority over the following functions:

- `init()`: init the basic settings of the contract;
- `setSuperChargerVault()`: change the `superChargerVault` of the contract;
- `setLender()`: manage the state variable `isLender`;
- `inCaseTokenGotStuck()`: transfer stuck eth or ERC20 token in the contract.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority, change the configuration of the contract, and transfer assets of the contract.

In the contract `WooLendingManager`, the role `isLender` has authority over the following functions:

- `setInterestRate()`: set the interest rate;
- `borrow()`: borrow assets from the `superChargerVault`;
- `repayWeekly()`: pay off a `weeklyRepayAmount` of debt;
- `repayAll()`: pay off all the debt;
- `repay()`: pay of a specified amount of debt.

Any compromise to the `isLender` account may allow a hacker to take advantage of this authority, change the interest configuration of the contract, and transfer assets of the contract.

In the contract `WooLendingManager`, the role `superChargerVault` has authority over the following functions:

- `setRepayAmount()`: manage the state variable `weeklyRepayAmount`.

Any compromise to the `superChargerVault` account may allow a hacker to take advantage of this authority and change the configuration of the contract.

In the contract `WooSuperChargerVault`, the role `_owner` has authority over the following functions:

- `init()`: init the basic settings of the contract;

- `setTreasury()`: change the `treasury` of the contract;
- `setInstantWithdrawFeeRate()`: manage the state variable `instantWithdrawFeeRate`;
- `inCaseTokenGotStuck()`: transfer stuck eth or ERC20 token in the contract.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority, change the configuration of the contract, and transfer assets of the contract.

In the contract `WooSuperChargerVault`, the role `onlyAdmin` has authority over the following functions:

- `startWeeklySettle()`: set the interest rate;
- `endWeeklySettle()`: borrow assets from the `superChargerVault`;

Any compromise to the `onlyAdmin` account may allow a hacker to take advantage of this authority and start/end the weekly settlement.

In the contract `WooSuperChargerVault`, the role `lendingManager` has authority over the following functions:

- `borrowFromLender()`: borrow assets to the lender;
- `repayFromLender()`: pay off the debt.

Any compromise to the `lendingManager` account may allow a hacker to take advantage of this authority and transfer assets of the contract.

In the contract `WooWithdrawManager`, the role `_owner` has authority over the following functions:

- `init()`: init the basic settings of the contract.
- `inCaseTokenGotStuck()`: transfer stuck eth or ERC20 token in the contract.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority, change the configuration of the contract, and transfer assets of the contract.

In the contract `WooWithdrawManager`, the role `onlyAdmin` has authority over the following functions:

- `setSuperChargerVault()`: manage the state variable `superChargerVault`.

Any compromise to the `isLender` account may allow a hacker to take advantage of this authority and change the `superChargerVault` of the contract.

In the contract `WooWithdrawManager`, the role `superChargerVault` has authority over the following functions:

- `addWithdrawAmount()`: transfer assets to the current contract and change the state variable `withdrawAmount`.

Any compromise to the `superChargerVault` account may allow a hacker to take advantage of this authority and increase a user's withdrawable amount.

The following content is based on commit 7bbb8a6971e785102ebaf5d308fe812711025baa.

In the contract `WooLendingManager`, the role `_owner` has authority over the newly added functions:

- `setWooPP()`: manage the state variable `wooPP`.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and change the configuration of the contract.

In the contract `WooSuperChargerVault`, the role `_owner` has authority over the new added functions:

- `migrateReserveVault()`: migrate assets from old vault to new vault.

Any compromise to the `_owner` account may allow a hacker to take advantage of this authority and transfer assets of the contract.

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 team acknowledged this issue and will ensure the owner/admin address uses multi-sig wallets.

EAR-02 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	WooLendingManager.sol: 88; WooSuperChargerVault.sol: 109, 121, 248, 256, 296; WooWithdrawManager.sol: 86, 112	Acknowledged

Description

The contract `WooLendingManager`, `WooSuperChargerVault`, `WooWithdrawManager` are serving as the underlying entity to interact with third-party `IWooAccessManager`, `IVaultV2`, `IWETH` protocols. 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 understand that the business logic of these contracts requires interaction with `IWooAccessManager`, `IVaultV2`, `IWETH`, etc. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[WOOFI]: The dependencies (e.g. IWooAccessManager, VaultV2, IWETH) are all from WooFi contracts, and have been audited and used in production for several months, w/o any safety incidences.

EAR-03 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	WooLendingManager.sol: 73, 104; WooSuperChargerVault.sol: 330; WooWithdrawManager.sol: 72, 97	Acknowledged

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected error.

Alleviation

[WOOFi]: The team acknowledged this issue and decided not to change the codebase this time. Double-checked that those address setters are all restricted as 'onlyOnwer'. Our admin will make sure the param is the non-zero address. Also, it can be reset at any time.

EAR-04 | Unknown Imported Source File

Category	Severity	Location	Status
Logical Issue	Informational	WooLendingManager.sol: 38~45, 48~49; WooSuperChargerVault.sol: 38~46, 48~51; WooWithdrawManager.sol: 38~49	Acknowledged

Description

The implementations of **linked** imported source files in each contract are unknown.

Recommendation

Recommend checking if the implementation of the imported source files meet the design. These **linked** imported files are out of the audit scope.

Alleviation

[WOOFi]: The team acknowledged this issue and decided not to change the codebase this time. The linked files are openzepplin libs (stable versions), and pretty safe to depend on.

EAR-05 | Missing Error Messages

Category	Severity	Location	Status
Coding Style	Informational	WooLendingManager.sol: 154, 182, 188, 207; WooSuperChargerVault.sol: 122, 244, 278, 287, 288, 298, 321; WooWithdrawManager.sol: 121	Acknowledged

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

Consider providing a string message to contain details about the error.

Alleviation

[WOOFi]: The team acknowledged this issue and decided not to change the codebase this time. Tenderly is used for debugging and error stack tracing; No need to set the error message for requiring anymore; and having the error message costs extra gas, and won't help more in debugging here.

EAR-06 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	WooLendingManager.sol: 73, 104, 108, 148, 153, 165, 169, 174, 181, 206; WooSuperChargerVault.sol: 112, 243, 253, 320, 330, 334; WooWithdrawManager.sol: 72, 97, 120	Partially Resolved

Description

Sensitive actions and the functions that affect the status of sensitive variables or transfer assets should be able to emit events.

Recommendation

Consider adding events for sensitive actions, and emit them in the function.

Alleviation

[WOOFi]: The team partially resolved this issue by adding missing events in commit 7bbb8a6971e785102ebaf5d308fe812711025baa. A few more events just added for WooLendingManager. For other contracts, the necessary events are already implemented.

```
// Contract WooLendingManager added events
event Borrow(address indexed user, uint256 assets);
event Repay(address indexed user, uint256 assets);
event InterestRateUpdated(address indexed user, uint256 oldInterest, uint256 newInterest);
```

WLM-01 | Divide Before Multiply

Category	Severity	Location	Status
Mathematical Operations	Informational	WooLendingManager.sol: 142	Resolved

Description

In the function `accureInterest()` of `WooLendingManager`, performing integer division before multiplication may truncate the low bits, losing the precision of calculation.

```
uint256 interest = borrowedPrincipal.mul(interestRate).div(10000).mul(duration).div(31536000);
```

Recommendation

We recommend applying multiplication before division to avoid loss of precision.

Alleviation

[WOOFi]: The team resolved this issue by applying multiplication before division in commit 7bbb8a6971e785102ebaf5d308fe812711025baa.

WSC-01 | Fee Collectors

Category	Severity	Location	Status
Centralization / Privilege	**Medium**	WooSuperChargerVault.sol: 185, 188	Acknowledged

Description

In the contract `WooSuperChargerVault`, there is a fee collector `treasury`, which gathers a fee on each call to `instantWithdraw()` at the fee rate of `instantWithdrawFeeRate`(default 30) out of 10000, over time, the account would gain many 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

[WOOFi]: The team acknowledged this issue and decided not to change the codebase this time. The fee collector address will be ensured to use the multi-sig wallet in production.

WSC-02 | Unused `Pausable` Feature

Category	Severity	Location	Status
Logical Issue, Volatile Code	Minor	WooSuperChargerVault.sol: 56	Resolved

Description

The parent contract `Pausable` of `WooSuperChargerVault` has no `public` or `external` functions to pause/unpause the contract, and the contract `WooSuperChargerVault` does not implement these two features either. Is the `Pausable` useless, or is it missing these two features?

Recommendation

We recommend checking if the code meets the design requirements.

Alleviation

[WOOFi]: The team resolved this issue by adding the related code in commit 7bbb8a6971e785102ebaf5d308fe812711025baa.

WSC-03 | Incompatibility With Deflationary Tokens

Category	Severity	Location	Status
Volatile Code	Minor	WooSuperChargerVault.sol: 139	Acknowledged

Description

When transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged transaction fee. For example, if a user deposits 100 deflationary tokens (with a 10% transaction fee), only 90 tokens actually arrived in the contract. However, the user can still withdraw 100 tokens from the contract, which causes the contract to lose 10 tokens in such a transaction. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies.

Recommendation

We advise the client to regulate the `want` token supported and add necessary mitigation mechanisms to keep track of accurate balances if there is a need to support deflationary tokens.

Alleviation

[WOOFi]: Irrelevant. The supercharger vaults and the `want` token are not permissionless. In fact, only the admin can create and set up the vaults. Supercharger vaults only apply to mainstream tokens, which are not deflationary. No need for a redundant checking balance, which is more gas-consuming.

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.

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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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.