



Security Assessment

Pinecone Finance

Sept 28th, 2021

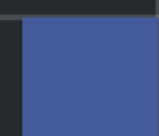
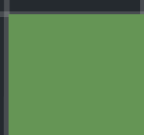


Table of Contents

Summary

Overview

[Project Summary](#)

[Audit Summary](#)

[Vulnerability Summary](#)

[Audit Scope](#)

Findings

[PFC-01 : Unknown implementations](#)

[PFC-02 : Lack of input validation](#)

[PFC-03 : Division Before Multiplication](#)

[PFC-04 : Centralization Risk](#)

[PFC-05 : Missing Emit Events](#)

[PFC-06 : Lack of input validation](#)

[PFC-07 : Rule of Calculation of Daily Reward](#)

[PFC-08 : Lack of `allocPCTPoint` Update](#)

[VPC-01 : Lack of input validation](#)

[VPC-02 : Centralization Risk](#)

[VPC-03 : Unknown implementations](#)

[VPC-04 : Fee Charging](#)

[VPC-05 : Redundant Functions](#)

[VPC-06 : Rule of Calculation of Daily Reward](#)

[VPT-01 : Lack of input validation](#)

[VPT-02 : Centralization Risk](#)

[VPT-03 : Unknown implementations](#)

[VPT-04 : Redundant Functions](#)

Appendix

Disclaimer

About

Summary

This report has been prepared for WBTECH.PTE.LTD to discover issues and vulnerabilities in the source code of the Pinecone Finance 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	Pinecone Finance
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/pinecone-finance/pinecone-contracts
Commit	ec2afdc1a47dbd5bf58a7b202952a6ae38353197 6da02c759251239e692bb6d368d6244facdf5a45

Audit Summary

Delivery Date	Sept 28, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

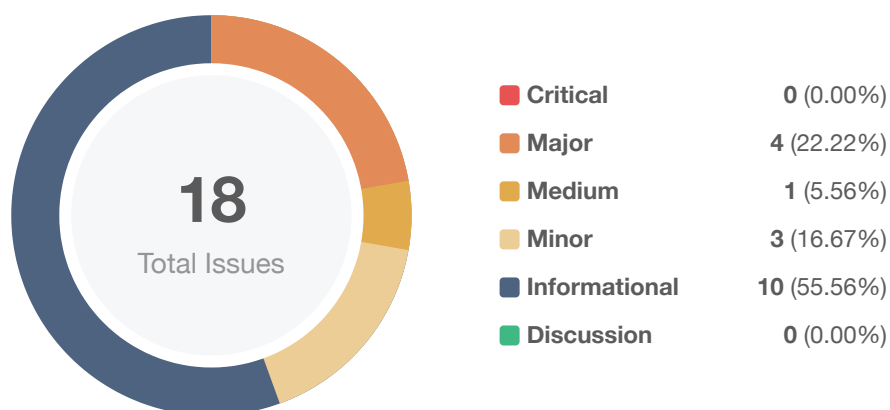
Vulnerability Level	Total	⚠ Pending	⊗ Declined	ℹ Acknowledged	🕒 Partially Resolved	✅ Resolved
🔴 Critical	0	0	0	0	0	0
🟠 Major	4	0	0	3	1	0
🟡 Medium	1	0	0	0	1	0
🟠 Minor	3	0	0	3	0	0
🟡 Informational	10	0	0	3	0	7
🟢 Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
PFC	PineconeFarm.sol	bebe49614ea81925f5aee758641f5fb84f820f76e567b0bac2b49e17ca13cc70
VPC	VaultPCT.sol	6301d9b1de6ce570282014c6ae0179bcacee68fc00045c65b6a4afe49da23edb
VPT	VaultPCTPair.sol	0a87566c9220d6a5aa458f4196efbda17d5a129e2dd5342975dd47a6243831c4

It is noted that the centralization risk of the contract `PineconeFarm.sol` and dependencies on other unknown contracts.

Findings



ID	Title	Category	Severity	Status
PFC-01	Unknown implementations	Compiler Error	Minor	ⓘ Acknowledged
PFC-02	Lack of input validation	Gas Optimization	Informational	✅ Resolved
PFC-03	Division Before Multiplication	Language Specific	Informational	✅ Resolved
PFC-04	Centralization Risk	Centralization / Privilege	Major	ⓘ Acknowledged
PFC-05	Missing Emit Events	Coding Style	Informational	✅ Resolved
PFC-06	Lack of input validation	Volatile Code	Informational	✅ Resolved
PFC-07	Rule of Calculation of Daily Reward	Logical Issue	Informational	✅ Resolved
PFC-08	Lack of <code>allocPCTPoint</code> Update	Logical Issue	Medium	⚠️ Partially Resolved
VPC-01	Lack of input validation	Gas Optimization	Informational	✅ Resolved
VPC-02	Centralization Risk	Centralization / Privilege	Major	ⓘ Acknowledged
VPC-03	Unknown implementations	Compiler Error	Minor	ⓘ Acknowledged
VPC-04	Fee Charging	Centralization / Privilege	Major	⚠️ Partially Resolved
VPC-05	Redundant Functions	Gas Optimization	Informational	ⓘ Acknowledged
VPC-06	Rule of Calculation of Daily Reward	Logical Issue	Informational	ⓘ Acknowledged
VPT-01	Lack of input validation	Gas Optimization	Informational	✅ Resolved
VPT-02	Centralization Risk	Centralization / Privilege	Major	ⓘ Acknowledged

ID	Title	Category	Severity	Status
VPT-03	Unknown implementations	Compiler Error	● Minor	ⓘ Acknowledged
VPT-04	Redundant Functions	Gas Optimization	● Informational	ⓘ Acknowledged

PFC-01 | Unknown implementations

Category	Severity	Location	Status
Compiler Error	● Minor	PineconeFarm.sol: 472, 7, 47	📄 Acknowledged

Description

The implementations of the `IPineconeToken` interface and the `IPineconeTokenCallee` interface in this protocol are unknown.

Examples:

```
123  pctPairAddress = IPineconeToken(_pctAddress).pctPair();
```

```
167  if (IPineconeToken(pctAddress).isMinter(address(this)) == false) {
```

The scope of the audit treats the unknown implementation as black boxes.

Recommendation

We understand that the business logic of this protocol requires interaction with `IPineconeToken` the `IPineconeTokenCallee` interface. But the development team should ensure the correctness of the unknown implementations.

Alleviation

[Pinecone Team]: Their implementations are in the contract `PineconeToken.sol`.

PFC-02 | Lack of input validation

Category	Severity	Location	Status
Gas Optimization	● Informational	PineconeFarm.sol: 122	🟢 Resolved

Description

Missing input validation in `initialize` for variable `_pctAddress`.

Recommendation

Consider adding validation check for `_pctAddress`.

Alleviation

The development team heeded our advice and resolved this issue in commit `6da02c759251239e692bb6d368d6244facdf5a45`.

PFC-03 | Division Before Multiplication

Category	Severity	Location	Status
Language Specific	● Informational	PineconeFarm.sol: 197	✓ Resolved

Description

Mathematical operations in the aforementioned function perform divisions before multiplications. Performing multiplication before division can sometimes avoid loss of precision.

Recommendation

We recommend applying multiplications before divisions if integer overflow would not happen.

```
uint256 earned1 = PCTPerBlock.mul(pool.allocPCTPoint).mul(28800).div(totalPCTAllocPoint);
```

Alleviation

The development team heeded our advice and resolved this issue in commit 6da02c759251239e692bb6d368d6244facdf5a45.

PFC-04 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	PineconeFarm.sol: 154~157, 186, 231~236, 258~262, 277, 295, 303~305, 178, 182, 309~311, 315, 591, 595, 599, 851, 649, 793, 806	① Acknowledged

Description

In the contract `PineconeFarm`, the role `owner` has the authority over the following functions:

- `setMinter(address _minter, bool _canMint)`
- `setCalDuration(uint256 _duration)`
- `add(uint256 _allocPCTPoint, address _want, bool _withUpdate, address _strat)`
- `set(uint256 _pid, uint256 _allocPCTPoint, bool _withUpdate)`
- `setPctPerProfitBNB(uint256 _pctPerProfitBNB)`
- `setPctPerBlock(uint256 _PCTPerBlock, uint256 _startBlock)`
- `setCakeRewardsPid(uint256 _cakeRewardsPid)`
- `inCaseTokensGetStuck(address _token, uint256 _amount)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- set minter and the minter can mint tokens to anyone.
- set duration
- add pool
- update pool
- set bnb profit
- set reward per block
- set cake rewards pid
- transfer current contract's PCT tokens to the owner

In the contract `PineconeFarm`, the role `dev` has the authority over the following functions:

- `setAuthContract(address _contract, bool _auth)`
- `setClaimCoolDown(uint256 _duration)`
- `setCakeRewardsNewPid(uint256 _cakeRewardPid)`
- `setPCTStakingPid(uint256 _pid)`
- `claimPCTPairDustBNB(address _to)`

- `claimPCTDustBNB(address _to)`
- `claimDeadDustBNB(address _to)`
- `migrateCakeRewardsPool(uint256 fromId, uint toId, bool newPool)`

Any compromise to the `dev` account may allow the hacker to take advantage of this and do the following:

- set whitelist for the contract
- set cool down duration
- change cake reward pool id
- change PCT Token pool id
- claim rewards
- migrate cake reward pool

In the contract `PineconeFarm`, the role `minter` has the authority over the following functions:

- `mintForProfit(address _to, uint256 _Profit, bool updatePCTRewards)`
- `stakeRewardsTo(address _to, uint256 _amount)`
- `mintForPresale(address _to, uint256 _amount)`

Any compromise to the `minter` account may allow the hacker to take advantage of this and do the following:

- mint tokens to the `to` account and the team account
- set duration
- mint tokens to the `to` account and the team account for pre-sale

Recommendation

We advise the client to carefully manage the `owner`, `dev`, and `minter` accounts' private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

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

Alleviation

The development team has removed the function `claimPCTPairDustBNB`, `claimPCTDustBNB` and `claimDeadDustBNB` in commit 6da02c759251239e692bb6d368d6244facdf5a45. They also replied that they will use a timelock and multi-signature wallets in the future.

PFC-05 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	● Informational	PineconeFarm.sol: 47	🟢 Resolved

Description

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

- `setMinter(address _minter, bool _canMint)`
- `setCalDuration(uint256 _duration)`
- `add(uint256 _allocPCTPoint, address _want, bool _withUpdate, address _strat)`
- `set(uint256 _pid, uint256 _allocPCTPoint, bool _withUpdate)`
- `setPctPerProfitBNB(uint256 _pctPerProfitBNB)`
- `setPctPerBlock(uint256 _PCTPerBlock, uint256 _startBlock)`
- `setCakeRewardsPid(uint256 _cakeRewardsPid)`
- `setAuthContract(address _contract, bool _auth)`
- `setClaimCoolDown(uint256 _duration)`
- `setCakeRewardsNewPid(uint256 _cakeRewardPid)`
- `setPCTStakingPid(uint256 _pid)`

Recommendation

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

Alleviation

The development team heeded our advice and resolved this issue in commit `6da02c759251239e692bb6d368d6244facdf5a45`.

PFC-06 | Lack of input validation

Category	Severity	Location	Status
Volatile Code	● Informational	PineconeFarm.sol: 285~293	✓ Resolved

Description

Missing input validation in `setDevAddress` and `setTeamRewardsAddress` for the variable `_addr`.

Recommendation

Consider adding validation check for `_addr`.

```
function setDevAddress(address _addr) external {
    require(devAddress == msg.sender, "no auth");
    require(_addr != address(0x0), "zero address!");
    devAddress = _addr;
}

function setTeamRewardsAddress(address _addr) external {
    require(teamRewardsAddress == msg.sender, "no auth");
    require(_addr != address(0x0), "zero address!");
    teamRewardsAddress = _addr;
}
```

Alleviation

The development team heeded our advice and resolved this issue in commit `6da02c759251239e692bb6d368d6244facdf5a45`.

PFC-07 | Rule of Calculation of Daily Reward

Category	Severity	Location	Status
Logical Issue	● Informational	PineconeFarm.sol: 711~715, 725~729	🟢 Resolved

Description

Could you detail the rule of calculation of daily reward? Especially the logical behind the if-else branch as shown below:

```
711     if (cap <= SEC_PER_DAY) {  
712         rewardAmt = pctTokenReward.accAmount;  
713     } else {  
714         rewardAmt = pctTokenReward.accAmount.mul(SEC_PER_DAY).div(cap);  
715     }
```

Alleviation

The development team deleted those codes in commit 6da02c759251239e692bb6d368d6244facdf5a45.

PFC-08 | Lack of `allocPCTPoint` Update

Category	Severity	Location	Status
Logical Issue	Medium	PineconeFarm.sol: 851	Partially Resolved

Description

Lack of updating `totalPCTAllocPoint` and the pools' `allocPCTPoint` after migrating the pool.

Alleviation

[Pinecone Team]: The migrated pools have `allocPoint` as zero.

VPC-01 | Lack of input validation

Category	Severity	Location	Status
Gas Optimization	● Informational	VaultPCT.sol: 30	✓ Resolved

Description

Missing input validation in `initialize` for variable `_config`.

Recommendation

Consider adding validation check for `_config`.

Alleviation

The development team heeded our advice and resolved this issue in commit `6da02c759251239e692bb6d368d6244facdf5a45`.

VPC-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	VaultPCT.sol: 114~119, 139, 178, 215, 225, 238, 109	ⓘ Acknowledged

Description

In the contract `VaultPCT`, the role `owner` has the authority over the following functions:

- `deposit(uint256 _wantAmt, address _user)`
- `withdrawAll(address _user)`
- `withdraw(uint256 _wantAmt, address _user)`
- `claim(address _user)`
- `claimBNB(address _user)`
- `updateCakeRewards(uint256 _amount)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- deposit owner's tokens to this contract and add user's reward records.
- withdraw the user's staking tokens.
- withdraw the user's staking tokens.
- do nothing
- update the user's reward and the user will not receive the unpaid reward
- update reward calculation factors such as `accAmount`, `totalAmount` and `accPerShare`.

In the contract `VaultPCT`, the role `dev` has the authority over the following functions:

- `setCalDuration(uint256 _duration)`

Any compromise to the `dev` account may allow the hacker to take advantage of this and do the following:

- set `calcDuration`

Recommendation

We advise the client to carefully manage the `owner` and `dev` accounts' private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to

be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

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

Alleviation

The development team replied that the `owner` will be the contract `PineconeFarm`. Anyone who wants to use this protocol should make sure the owner of this contract is the contract `PineconeFarm`.

VPC-03 | Unknown implementations

Category	Severity	Location	Status
Compiler Error	● Minor	VaultPCT.sol: 5~6	ⓘ Acknowledged

Description

There are several unknown implementations of the `VaultBase` interface and the `Strat` interface in this protocol.

Examples:

```
30     _VaultBase_init(_config, address(0));
31     _Strat_init(config.PCT(), address(0));
```

The scope of the audit treats the unknown implementation as black boxes.

Recommendation

We understand that the business logic of this protocol requires interaction with `VaultBase` and `Strat`. But the development team should ensure the correctness of the unknown implementation.

Alleviation

[Pinecone Team]: The two functions in the description are implemented in contracts `VaultBase.sol` and `Strat.sol`.

VPC-04 | Fee Charging

Category	Severity	Location	Status
Centralization / Privilege	● Major	VaultPCT.sol: 200, 156	🕒 Partially Resolved

Description

There is a fee-charging from the `_user` account to the `devAddress` account when calling the `withdrawAll(address _user)` and `withdraw(uint256 _wantAmt, address _user)` functions. Are that designed as expected?

Alleviation

[Pinecone Team]: There is a fee-charging if a user withdrawing in 3 days.

VPC-05 | Redundant Functions

Category	Severity	Location	Status
Gas Optimization	● Informational	VaultPCT.sol: 101, 215	ⓘ Acknowledged

Description

Functions `pendingRewardsValue()` and `claim(address _user)` return nothing but constant values. Are that designed as expected?

Recommendation

Consider deleting the functions if they are useless.

Alleviation

[Pinecone Team]: They are used by the external functions.

VPC-06 | Rule of Calculation of Daily Reward

Category	Severity	Location	Status
Logical Issue	● Informational	VaultPCT.sol: 263~267	ⓘ Acknowledged

Description

Could you explain the rule of calculation of daily reward? Especially the logical behind the if-else branch as shown below:

```
263     if (cap <= SEC_PER_DAY) {  
264         return cakeTokenReward.accAmount;  
265     } else {  
266         return cakeTokenReward.accAmount.mul(SEC_PER_DAY).div(cap);  
267     }
```

Alleviation

[Pinecone team]: To calculate the average income for a period.

VPT-01 | Lack of input validation

Category	Severity	Location	Status
Gas Optimization	● Informational	VaultPCTPair.sol: 19~20	🟢 Resolved

Description

Missing input validation in `initialize` for variables `_config` and `_stakingToken`.

Recommendation

Consider adding validation check for `_config` and `_stakingToken`.

Alleviation

The development team heeded our advice and resolved this issue in commit `6da02c759251239e692bb6d368d6244facdf5a45`.

VPT-02 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Major	VaultPCTPair.sol: 89, 109, 134, 158	ⓘ Acknowledged

Description

In the contract `VaultPCTPair`, the role `owner` has the authority over the following functions:

- `deposit(uint256 _wantAmt, address _user)`
- `withdrawAll(address _user)`
- `withdraw(uint256 _wantAmt, address _user)`
- `claim(address _user)`

Any compromise to the `owner` account may allow the hacker to take advantage of this and do the following:

- deposit owner's tokens to this contract and add user's reward records.
- transfer staking tokens to anyone
- transfer staking tokens to anyone
- do nothing

Recommendation

We advise the client to carefully manage the `owner` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

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

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

Alleviation

The development team replied that the `owner` will be the contract `PineconeFarm`. Anyone who wants to use this protocol should make sure the owner of this contract is the contract `PineconeFarm`.

VPT-03 | Unknown implementations

Category	Severity	Location	Status
Compiler Error	● Minor	VaultPCTPair.sol: 5~6	ⓘ Acknowledged

Description

There are several unknown implementations of the `VaultBase` interface and the `Strat` interface in this protocol.

Examples:

```
30     _VaultBase_init(_config, address(0));
31     _Strat_init(config.PCT(), address(0));
```

The scope of the audit treats the unknown implementation as black boxes.

Recommendation

We understand that the business logic of this protocol requires interaction with `VaultBase` and `Strat`. But the development team should ensure the correctness of the unknown implementation.

Alleviation

[Pinecone Team]: The two functions in the description are implemented in contracts `VaultBase.sol` and `Strat.sol`.

VPT-04 | Redundant Functions

Category	Severity	Location	Status
Gas Optimization	● Informational	VaultPCTPair.sol: 74, 79, 83, 32	ⓘ Acknowledged

Description

Functions `earned1Address()`, `earnedOf(address _user)`, `pendingRewardsValue()`, `pendingRewards(address _user)`, and `claim(address _user)` return nothing but constant values. Are that designed as expected?

Recommendation

Consider deleting the functions if they are useless.

Alleviation

[Pinecone Team]: They are used by the external functions.

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.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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.

