CERTIK

Code Security Assessment

Hamster

Jan 26th, 2022

Table of Contents

Summary

This report has been prepared for Hamster to discover issues and vulnerabilities in the source code of the Hamster 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	Hamster
Description	HamsterSwap is a defi protocol operating on the BSC (Binance Smart Chain) focused on Yield Farming.
Platform	other
Language	Solidity
Codebase	Initial audit: The client provided the source code in a Zip file Updated code: • CHEESE 0x045d9E82262dDa9257dF90bC30De74ef6573521a • Syrup 0xD9745C632d76199c9D684D3DCF98Da1Bcfb4dD5E • MasterChef 0x2D178D981AFeD27c492421C93C68267a103fA578 • Presale 0x56C9E0311EED40EeE8Ee182DFb6fd247DF483a28
Commit	

Audit Summary

Delivery Date	Jan 26, 2022
Audit Methodology	Static Analysis, Manual Review

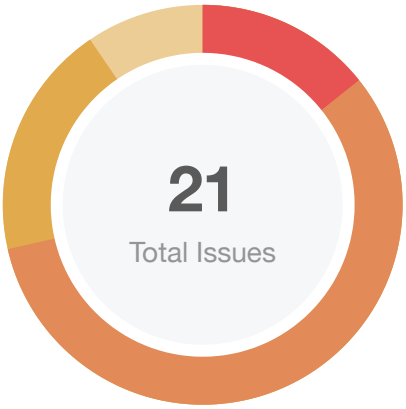
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
Critical	3	0	0	2	0	0	1
Major	12	0	0	11	0	0	1
Medium	4	0	0	1	0	0	3
Minor	2	0	0	2	0	0	0
Informational	0	0	0	0	0	0	0
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
CHE	CHEESE.sol	c62ed3e529b6ce0cdd7fa71326266c63b46ebed6e580569012f7f7ecb18a7a97
HSC	masterchef.sol	1a2e520c039f40065af0731d1a6184cf8a336224de1513b7a27db839972f7eb8
HSK	presale.sol	c41ce24bfe3a00a772cd7b58a0465cf29a934e81a4d4b5c76b16052ba33688f4

Findings



Critical	**3** (14.29%)
Major	**12** (57.14%)
Medium	**4** (19.05%)
Minor	**2** (9.52%)
Informational	**0** (0.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CHE-01	Privileged functions allow owner to mint unlimited `Cheese` tokens and withdraw BNB tokens	**Centralization / Privilege**	**Major**	Acknowledged
CHE-02	Voting Power Not Moved Along With token transfer/mint/burn	Logical Issue	Major	Acknowledged
HSC-01	OpenZeppelin library code included in source code	Volatile Code	Medium	Acknowledged
HSC-02	Redundant contract `Cheese`	Volatile Code	Major	Acknowledged
HSC-03	Usage of `transfer()` for sending Ether	Volatile Code	Minor	Acknowledged
HSC-04	Privileged functions allow owner to mint/burn SyrupBar tokens at will and withdraw Cheese/BNB tokens	**Centralization / Privilege**	**Major**	Acknowledged
HSC-05	Incorrect calling of `_moveDelegates()`	Logical Issue	Major	Acknowledged
HSC-06	Voting Power Not Moved Along With token transfer	Logical Issue	Major	Acknowledged
HSC-07	Third Party Dependencies	Volatile Code	Minor	Acknowledged
HSC-08	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
HSC-09	SafeMath Not Used	Mathematical Operations	Medium	Resolved
HSC-10	Error-prone decimal calculation	Logical Issue	Major	Acknowledged
HSC-11	Logic Flaw In `emergencyWithdraw()`	Logical Issue	Critical	Resolved

ID	Title	Category	Severity	Status
HSC-12	Over Minted Token	Logical Issue	Medium	Resolved
HSC-13	add() Function Not Restricted	Volatile Code	Major	Resolved
HSC-14	Incompatibility With Deflationary Tokens(Farming)	Volatile Code	Major	Acknowledged
HSC-15	Owner can take away all stacked tokens with the `migrate` function	**Centralization / Privilege**	**Critical**	Acknowledged
HSC-16	Owner can take away all stacked tokens with the `saftApprove` function	**Centralization / Privilege**	**Critical**	Acknowledged
HSK-01	Privileged functions allow owner to withdraw tokens	**Centralization / Privilege**	**Major**	Acknowledged
HSK-02	Lack of input validation and incorrect calculation	Logical Issue	Major	Acknowledged
HSK-03	`endPresale()` does not really end presale	Logical Issue	Medium	Resolved

CHE-01 | Privileged functions allow owner to mint unlimited `Cheese` tokens and withdraw BNB tokens

Category	Severity	Location	Status
Centralization / Privilege	**Major**	CHEESE.sol: 1088, 859	Acknowledged

Description

In the contract `[CHEESE.sol]`, the role `[owner]` has the authority over the following function:

- [mint()]
- [recoverBNB()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and mint unlimited Cheese tokens, steal BNB tokens from the contract.

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 are some feasible suggestions that would also mitigate the potential risk at the different levels in terms 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 compromised;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

No response from the client regarding this finding.

CHE-02 | Voting Power Not Moved Along With token transfer/mint/burn

Category	Severity	Location	Status
Logical Issue	Major	CHEESE.sol: 861	Acknowledged

Description

The `_moveDelegates()` is not called in `_mint()`, `_burn()`, `_transfer()`. So the voting power is not moved when tokens are minted/burned/transferred.

Recommendation

We advise the client to call `_moveDelegates()` when tokens are minted/burned/transferred.

Alleviation

No response from the client regarding this finding.

HSC-01 | OpenZeppelin library code included in source code

Category	Severity	Location	Status
Volatile Code	Medium	masterchef.sol: 6~950 CHEESE.sol: 5~854 presale.sol: 6~571	Acknowledged

Description

It is highly recommended NOT to include OpenZeppelin library code directly in source code, but import the original library to minimize risk because even slight changes to the library code may lead to critical/major vulnerabilities/bugs. For Solidity version 0.6.x, the latest OpenZeppelin version 3.4.2 should be used.

Recommendation

We advise the client to remove OpenZeppelin library code from source code and import the latest OpenZeppelin version.

Alleviation

No response from the client regarding this finding.

HSC-02 | Redundant contract `Cheese`

Category	Severity	Location	Status
Volatile Code	Major	CHEESE.sol: 857 masterchef.sol: 952~1194	Acknowledged

Description

The contract `Cheese` exists in both `CHEESE.sol` and `masterchef.sol`. And it is the one in `CHEESE.sol` deployed at `0xdE4C794867Be3Ad6c3ddA8CB42Ecad46eb9Fab68` on BSC. See https://bscscan.com/address/0xdE4C794867Be3Ad6c3ddA8CB42Ecad46eb9Fab68#code. But the contract `MasterChef` references the `Cheese` contract in `masterchef.sol`, which means 2 different `Cheese` contracts are deployed on chain: one is used by outside users, one is used by contract `MasterChef`.

Recommendation

We advise the client to remove the redundant contract `Cheese` in `masterchef.sol`.

Alleviation

[From Hamster team]: We have used governance feature in the CHEESE token Here we have customized the mint function. So we can not use default IBEP20 interface. And the Cheese contract of the Masterchef contract is same with the deployed Cheese contract. We can skip this problem."

HSC-03 | Usage of `transfer()` for sending Ether

Category	Severity	Location	Status
Volatile Code	Minor	presale.sol: 686, 647, 694 masterchef.sol: 1461 CHEESE.sol: 1090	Acknowledged

Description

After EIP-1884 was included in the Istanbul hard fork, it is not recommended to use `.transfer()` or `.send()` for transferring ether as these functions have a hard-coded value for gas costs making them obsolete as they are forwarding a fixed amount of gas, specifically `2300`. This can cause problems in the case of transfering funds to other contracts instead of EOAs.

Recommendation

We advise the client to use [the `sendValue()` function] from the `Address.sol` in OpenZeppelin library.

Alleviation

[From Hamster team]: "We can not use sendValue function. If we use this, "member send value not found or not visible after argument-dependent lookup in address" error message appear"

HSC-04 | Privileged functions allow owner to mint/burn SyrupBar tokens at will and withdraw Cheese/BNB tokens

Category	Severity	Location	Status
Centralization / Privilege	Major	masterchef.sol: 1459, 1220, 1204, 1199	Acknowledged

Description

In the contract `[SyrupBar]` inside `masterchef.sol`, the role `[owner]` has the authority over the following function:

- [mint()]
- [burn()]
- [safeCheeseTransfer()]
- [recoverBNB()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and mint/burn SyrupBar tokens at will, steal Cheese/BNB tokens from the contract.

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 are some feasible suggestions that would also mitigate the potential risk at the different levels in terms 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 compromised;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

No response from the client regarding this finding.

HSC-05 | Incorrect calling of `_moveDelegates()`

Category	Severity	Location	Status
Logical Issue	Major	masterchef.sol: 1206	Acknowledged

Description

For burning tokens, voting power must be moved from `_delegates[_from]` to `address(0)`, NOT the other way around.

Recommendation

We advise the client to move voting power from `_delegates[_from]` to `address(0)`.

Alleviation

[From Hamster team]: "CAKE token is governance token what many user are using now without problem"

HSC-06 | Voting Power Not Moved Along With token transfer

Category	Severity	Location	Status
Logical Issue	Major	masterchef.sol: 1201	Acknowledged

Description

`_moveDelegates()` is NOT called in `_transfer()` or `transfer()`/`transferFrom()`, so voting power is NOT moved when tokens are transferred.

Recommendation

We advise the client to call `_moveDelegates()` in `_transfer()`.

Alleviation

No response from the client regarding this finding.

HSC-07 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	masterchef.sol: 1617 presale.sol: 637	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party [AggregatorV3Interface], [IMigratorChef] 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 [Presale], [MasterChef] requires interaction with [AggregatorV3Interface], [IMigratorChef]. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[From Hamster team]:"I think we can skip this problem. The AggregatorV3Interfact contract was used for getting the BNB price. I am reading only the BNB price from that. So there is no problem."

HSC-08 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	masterchef.sol: 1795, 1791, 1786, 1782, 1767, 1606, 1579, 1562, 1552	Acknowledged

Description

In the contract `[MasterChef]`, the role `[owner]` has the authority over the following function:

- [updateMultiplier()]
- [add()]
- [set()]
- [setMigrator()]
- [saftApprove()]
- [updateCheesePerBlock()]
- [recoverBNB()]
- [recoverBNBFromCheese()]
- [recoverBNBfromSyrup()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and make the contract malfunction, steal tokens from the contract and other contracts.

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 are some feasible suggestions that would also mitigate the potential risk at the different levels in terms 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 compromised;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

No response from the client regarding this finding.

HSC-09 | SafeMath Not Used

Category	Severity	Location	Status
Mathematical Operations	Medium	masterchef.sol: 1677, 1699	Resolved

Description

SafeMath from OpenZeppelin is not used in the following functions which makes them possible for overflow/underflow and will lead to an inaccurate calculation result.

- `deposit()`
- `withdraw()`

Recommendation

We advise the client to use OpenZeppelin's SafeMath library for all of the mathematical operations.

Reference: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/math/SafeMath.sol

Alleviation

Fixed at https://bscscan.com/address/0x2D178D981AFeD27c492421C93C68267a103fA578#code

HSC-10 | Error-prone decimal calculation

Category	Severity	Location	Status
Logical Issue	Major	masterchef.sol: 1708~1711, 1700~1702, 1685~1688, 1678	Acknowledged

Description

The `amount` parameter for erc20 `transfer()` and `transferFrom()` uses the decimals returned by erc20 `decimals()`, NOT fixed value 18. So if any erc20 token with decimals other than 18 is added to the pool, the identified code transfers an incorrect amount of tokens.

Recommendation

We advise the client to use proper decimals for erc20 token transfer.

Alleviation

[From Hamster team]: "We have implemented like that because we are using the Hamster token what decimals is 7. Otherwise we can not add Hamster Token for staking on our contract."

HSC-11 | Logic Flaw In `emergencyWithdraw()`

Category	Severity	Location	Status
Logical Issue	Critical	masterchef.sol: 1758	Resolved

Description

Please Note This Is Based On a Known Exploit. Use with Cautious!

When `msg.sender` calls `enterStaking()` , `syrup` token will be minted to `msg.sender` when `pool.lpToken` is staked in the contract. However, if the `msg.sender` calls `emergencyWithdraw()`, the `pool.lpToken` can be transferred back to the `msg.sender` but the `syrup` token that has been minted to the `msg.sender` will not be burnt. Therefore, `msg.sender` can call `enterStaking()` and `emergencyWithdraw()` repeatedly to ultimately mint a huge amount of syrup token, with just the same amount of pool.lpToken

Recommendation

We advise the client to burn the same amount of `syrup` along with the withdraw of `pool.lpToken` when calling the `emergencyWithdraw()`. i.e:

```
function emergencyWithdraw(uint256 _pid) public {
    PoolInfo storage pool = poolInfo[_pid];
    UserInfo storage user = userInfo[_pid][msg.sender];
    if(_pid == 0) {
        syrup.burn(msg.sender, user.amount);
    }
    uint256 amount = user.amount;
    user.amount = 0;
    user.rewardDebt = 0;
    pool.lpToken.safeTransfer(address(msg.sender), amount);
    emit EmergencyWithdraw(msg.sender, _pid, amount);
}
```

Alleviation

Fixed at https://bscscan.com/address/0x2D178D981AFeD27c492421C93C68267a103fA578#code

HSC-12 | Over Minted Token

Category	Severity	Location	Status
Logical Issue	Medium	masterchef.sol: 1663~1664	Resolved

Description

`updatePool()` function minted 100% + 10% (dev fee is included as 10% of the 100%) of total rewards.

Recommendation

Fix to mint 100% of the block reward instead of 100% + 10% (dev fee is included as 10% of the 100%) like in other MasterChef clones.

Alleviation

Fixed at https://bscscan.com/address/0x2D178D981AFeD27c492421C93C68267a103fA578#code

HSC-13 | add() Function Not Restricted

Category	Severity	Location	Status
Volatile Code	Major	masterchef.sol: 1562	Resolved

Description

The comment in line L1561, mentioned // XXX DO NOT add the same LP token more than once. Rewards will be messed up if you do.

The total amount of reward eggReward in function `updatePool()` will be incorrectly calculated if the same LP token is added into the pool more than once in function `add()`.

However, the code is not reflected in the comment behaviors as there isn't any valid restriction on preventing this issue.

The current implementation is relying on the trust of the owner to avoid repeatedly adding the same LP token to the pool, as the function will only be called by the owner.

Recommendation

Detect whether the given pool for addition is a duplicate of an existing pool. The pool addition is only successful when there is no duplicate. Using a mapping of `addresses` -> `booleans`, which can restricted the same address being added twice.

Alleviation

Fixed at https://bscscan.com/address/0x2D178D981AFeD27c492421C93C68267a103fA578#code

HSC-14 | Incompatibility With Deflationary Tokens(Farming)

Category	Severity	Location	Status
Volatile Code	Major	masterchef.sol	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 stakes 100 deflationary tokens (with a 10% transaction fee) in a MasterChef, 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.

The MasterChef takes the pool token balance(the `lpSupply`) into account when calculating the users' reward. An attacker can repeat the process of deposit and withdraw to lower the token balance(`lpSupply`) in a deflationary token pool and cause the contract to increase the reward amount.

Reference: https://thoreum-finance.medium.com/what-exploit-happened-today-for-gocerberus-and-garuda-also-for-lokum-ybear-piggy-caramelswap-3943ee23a39f

Recommendation

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

Alleviation

No response from the client regarding this finding.

HSC-15 | Owner can take away all stacked tokens with the `migrate` function

Category	Severity	Location	Status
Centralization / Privilege	**Critical**	masterchef.sol: 1611	Acknowledged

Description

In the contract `masterchef.sol`, the role `owner` has authority over the following functions:

- migrate()
- setMigrator()

Any compromise to the `owner` account may allow a hacker to take advantage of this authority and set a malicious Migrator to take away all stacked tokens.

Recommendation

It's recommended to remove the `migrate` function from the contract if the project does not require it.

Alleviation

No response from the client regarding this finding.

HSC-16 | Owner can take away all stacked tokens with the `saftApprove` function

Category	Severity	Location	Status
Centralization / Privilege	**Critical**	masterchef.sol: 1767~1769	ⓘ Acknowledged

Description

In the contract `masterchef.sol`, the role `owner` has authority over the following functions:

- saftApprove()

The owner can utilize this function to approve any address the max allowance of all stacked tokens. The address then can take away all tokens from the MasterChef contract with the "transferFrom" function.

Recommendation

It's recommended to remove the function from the contract if the project can't justify why such function is needed.

Alleviation

[From Hamster team:] "If some tokens arrive in the masterchef with wrong operation, we can withdraw them and send to users. Hamster contract received more than 200 BNB and now we can’t touch them. We wanna protect owner investors if they are something wrong."

HSK-01 | Privileged functions allow owner to withdraw tokens

Category	Severity	Location	Status
Centralization / Privilege	**Major**	presale.sol: 720, 716, 712, 708, 704, 699, 690	Acknowledged

Description

In the contract `[Presale]`, the role `[owner]` has the authority over the following function:

- [endPresale()]
- [withdrawTokens()]
- [saftApprove()]
- [setStartTime()]
- [setEndTime()]
- [setHardCap()]
- [setSoftCap()]

Any compromise to the `[owner]` account may allow the hacker to take advantage of this and make the contract malfunction, steal tokens from the contract.

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 are some feasible suggestions that would also mitigate the potential risk at the different levels in terms 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 compromised;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

No response from the client regarding this finding.

HSK-02 | Lack of input validation and incorrect calculation

Category	Severity	Location	Status
Logical Issue	Major	presale.sol: 726	Acknowledged

Description

In extreme conditions or when the data provider is hacked, the price value `answer` can be negative which is not handled properly by the code. And normally the actual price value is not an integer, so the `decimals()` from `AggregatorV3Interface` must be used for proper calculation.

Recommendation

We advise the client to revert when price value is negative and use `decimals()` from `AggregatorV3Interface` for proper calculation.

Alleviation

No response from the client regarding this finding.

HSK-03 | `endPresale()` does not really end presale

Category	Severity	Location	Status
Logical Issue	Medium	presale.sol: 690	Resolved

Description

When `block.timestamp >= startTime && block.timestamp <= endTime`, even after `endPresale()` is called, users can still successfully call `buyTokens()`.

Recommendation

We advise the client to either remove time checking in `hasEnded()` or require that `endPresale()` can only be called after presale period is over.

Alleviation

Fixed at https://bscscan.com/address/0x56C9E0311EED40EeE8Ee182DFb6fd247DF483a28#code

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.

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.