CERTIK

Security Assessment

Seedify finance Locked-Yield-Farming

CertiK Assessed on Nov 13th, 2023

CertiK Assessed on Nov 13th, 2023

Seedify finance Locked-Yield-Farming

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES	ECOSYSTEM	METHODS
Farming	EVM Compatible	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 11/13/2023	N/A

CODEBASE

https://github.com/Seedifyfund/Locked-Yield-Farming/blob/main/contracts/LockedFarming.sol

View All in Codebase Page

COMMITS

- def3af9ebf0f78af2235e3f98ff2e501cc118d2d
- 49e987095e999cee9602ea9421452162ef57a354
- 626d304d632b06ab8b660f83ceeab33b916b50a7

View All in Codebase Page

Highlighted Centralization Risks

Withdraws can be disabled

Vulnerability Summary

9	6	0	0	3	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

0	Critical		Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.
2	Major	2 Acknowledged	Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.
0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
5	Minor	4 Resolved, 1 Acknowledged	Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.
2	Informational	2 Resolved	Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS

SEEDIFY FINANCE LOCKED-YIELD-FARMING

CODEBASE | SEEDIFY FINANCE LOCKED-YIELD-FARMING

Repository

https://github.com/Seedifyfund/Locked-Yield-Farming/blob/main/contracts/LockedFarming.sol

Commit

- def3af9ebf0f78af2235e3f98ff2e501cc118d2d
- 49e987095e999cee9602ea9421452162ef57a354
- 626d304d632b06ab8b660f83ceeab33b916b50a7

AUDIT SCOPE | SEEDIFY FINANCE LOCKED-YIELD-FARMING

3 files audited ● 1 file with Acknowledged findings ● 2 files without findings

ID	Repo	File	SHA256 Checksum
● LFL	Seedifyfund/Locked-Yield-Farming	contracts/LockedFarming.sol	60a71cb909f31604a7b430fabdd6bba3ab46e1dd4fc4d02c9a5900873ce8b262
● LFY	Seedifyfund/Locked-Yield-Farming	LockedFarming.sol	2bf96c4cba09be2c4b45c29b5829a7fa2f5fc95eb0ffc4829c3a9ff44d1bfffa
● LFF	Seedifyfund/Locked-Yield-Farming	LockedFarming.sol	5b21ff0a0ee99da564314c7d135bddacdcc6716ae298b0fe0292e7b2f548efc2

APPROACH & METHODS

SEEDIFY FINANCE LOCKED-YIELD-FARMING

This report has been prepared for Seedify finance to discover issues and vulnerabilities in the source code of the Seedify finance Locked-Yield-Farming project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis 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:

- Testing the smart contracts against both common and uncommon attack vectors;
- 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.

FINDINGS | SEEDIFY FINANCE LOCKED-YIELD-FARMING

9	0	2	0	5	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Seedify finance Locked-Yield-Farming. Through this audit, we have uncovered 9 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

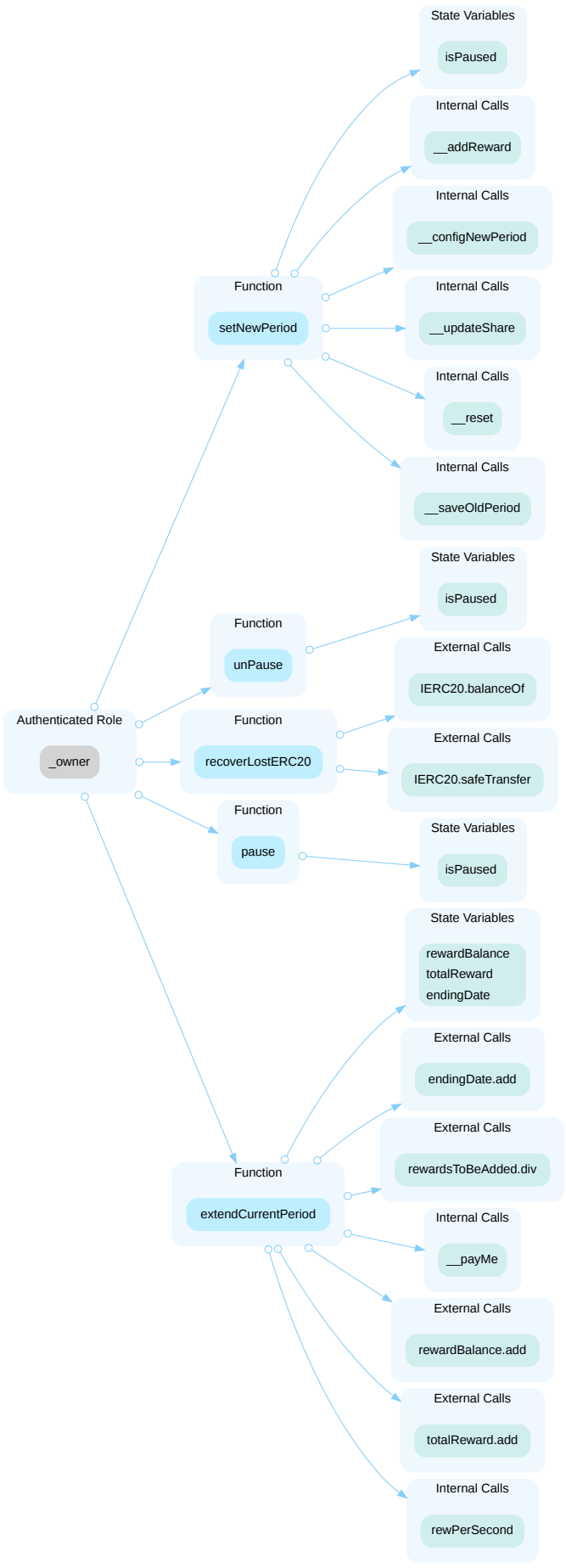
ID	Title	Category	Severity	Status
GLOBAL-02	**Centralization Risks In LockedFarming.Sol**	**Centralization**	**Major**	**Acknowledged**
GLOBAL-03	Incompatibility With Deflationary Tokens	Logical Issue	Major	Acknowledged
GLOBAL-01	Ambiguous Use Of `isPaused`	Design Issue, Inconsistency	Minor	Resolved
GLOBAL-04	Inconsistency In `initialStake` Behavior And Comment Description	Inconsistency	Minor	Resolved
GLOBAL-05	Delay In Updating Expired Periods' Rewards Details	Design Issue	Minor	Resolved
LFL-01	Third-Party Dependency Usage	Design Issue	Minor	Acknowledged
LFL-02	Potentially Limits Recovery Of Excess Tokens	Logical Issue	Minor	Resolved
LFL-03	Incomplete Information Retrieval From `userDeposits()`	Design Issue	Informational	Resolved
LFL-04	Flawed `stakedBalance` Update Mechanism In `__withdraw()` Function	Logical Issue	Informational	Resolved

GLOBAL-02 | CENTRALIZATION RISKS IN LOCKEDFARMING.SOL

Category	Severity	Location	Status
Centralization	**Major**		**Acknowledged**

Description

In the contract `SMD_v5` the role `_owner` has authority over the functions shown in the diagram below. Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set rewards periods, extend the current period, pause/unpause functions, and withdraw excess tokens mistakenly sent to the contract. Notably, the hacker can prevent users from withdrawing staked tokens by updating `lockDuration` to a large value via `setNewPeriod()`.


State Variables
isPaused
Internal Calls
__addReward
Internal Calls
__configNewPeriod
Function
setNewPeriod
Internal Calls
__updateShare
Internal Calls
__reset
Internal Calls
__saveOldPeriod
State Variables
isPaused
Function
unPause
External Calls
IERC20.balanceOf
Authenticated Role
_owner
Function
recoverLostERC20
External Calls
IERC20.safeTransfer
Function
pause
State Variables
isPaused
State Variables
rewardBalance
totalReward
endingDate
External Calls
endingDate.add
External Calls
rewardsToBeAdded.div
Function
extendCurrentPeriod
Internal Calls
__payMe
External Calls
rewardBalance.add
External Calls
totalReward.add
Internal Calls
rewPerSecond

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets. Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Seedify Finance Team, 10/24/2023]: Issue acknowledged. I won't make any changes for the current version.

GLOBAL-03 | INCOMPATIBILITY WITH DEFLATIONARY TOKENS

Category	Severity	Location	Status
Logical Issue	Major		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), only 90 tokens actually arrive 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.

If `tokenAddress` (the staking token) is deflationary, an attacker can repeatedly call `deposit()` and `withdraw()` / `emergencyWithdraw()` to empty the contract's funds and put other users' tokens at risk.

Similarly, if the `rewardTokenAddress` (the rewards token) is deflationary, the actual balance is lower than anticipated, leaving the potential for insufficient funds when disbursing rewards.

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

Recommendation

Is is recommended to either not use deflationary tokens or accurately account for the received amount post-fees, both for staking and rewards.

Alleviation

[Seedify Finance Team, 10/24/2023]: Issue acknowledged. I won't make any changes for the current version.

GLOBAL-01 | AMBIGUOUS USE OF `isPaused`

Category	Severity	Location	Status
Design Issue, Inconsistency	Minor		Resolved

Description

In the `SMD_v5` contract, certain functions require `isPaused` variable to be false, suggesting these functions may be paused under specific conditions. However, based on the current implementation, the contract will only ever be paused before the initial period is set. The sole function that can toggle `isPaused` to true is `__reset()`, which is called within `setNewPeriod()`. Immediately afterward, `setNewPeriod()` invokes `__setStartEnd()`, which in turn sets `isPaused` back to false. As a result, functions that require `isPaused` to be false will always proceed without interruption.

Also, the pause behavior is inconsistent. For example:

1. The `viewOldRewards()` function is a view function that requires `isPaused` to be false. However, the other view functions lack this requirement.
2. The `withdraw()` does not directly require `isPaused` to be false, instead, it relies on `viewOldRewards()` which does. This implementation leads to ambiguity, causing one to question what the intended design it.
3. The `emergencyWithdraw()` function does not require `isPaused` to be false, it is recommended to check if this aligns with the intended design.

Recommendation

It is recommended to review the usage of the `isPaused` mechanism to ensure it aligns with its intended behavior.

Alleviation

[Seedify Finance Team, 10/24/2023]: The team heeded the advice and resolved the issue in commit 49e987095e999cee9602ea9421452162ef57a354.

GLOBAL-04 INCONSISTENCY IN `initialStake` BEHAVIOR AND COMMENT DESCRIPTION

Category	Severity	Location	Status
Inconsistency	Minor		Resolved

Description

The `Deposits` struct maintains data related to the deposits made by a user. Within this struct, the `initialStake` field is intended to record the timestamp when a user renews their stake for new periods, as described in the comment. However, there seems to be a discrepancy. The `initialStake` gets updated even when a user re-stakes within the current period, diverging from its intended behavior as per the comment. As a result, users could be prevented from withdrawing their stakes, as withdrawals are permitted only after `lockDuration.mul(SECONDS_PER_HOUR)` from the `initialStake`."

```
73      /**
74
    * @notice struct which should represent the deposit made by a wallet based on
all period if the wallet
75       *         called {renew}.
76       *
77       * @param amount amount of LP {tokenAddress} deposited accross all period.
78
    * @param initialStake should be the timestamp at which the wallet renewed their
stake for new periods.
79
    * @param latestClaim latest timestamp at which the wallet claimed their
rewards.
80
    * @param userAccShare should be the amount of rewards per wei of deposited LP
token {tokenAddress}
81       *        accross all periods.
82
    * @param currentPeriod should be the lastest periodCounter at which the wallet
participated.
83       */
84      struct Deposits {
85          uint256 amount;
86          uint256 initialStake;
87          uint256 latestClaim;
88          uint256 userAccShare;
89          uint256 currentPeriod;
90      }
```

Recommendation

It is recommended to avoid updating the `initialStake` when a user restakes during the ongoing period. However, if a user's most recent stake was in a past period, their `initialStake` should be adjusted.

Alleviation

[Seedify Finance Team, 10/24/2023]: The team has modified the field name to `latestStakeAt`, which indicates in commit 49e987095e999cee9602ea9421452162ef57a354.

```
     * @param latestStakeAt timestamp at which the latest stake has been made by the
wallet for current
     *        period. Maturity date will be re-calculated from this timestamp which
means each time the
     *        wallet stakes a new amount it has to wait for `lockDuration` before
being able to withdraw.
```

GLOBAL-05 DELAY IN UPDATING EXPIRED PERIODS' REWARDS DETAILS

Category	Severity	Location	Status
Design Issue	Minor		Resolved

Description

The `endAccShare` mapping in the contract captures the details of expired periods that determine the rewards users can claim from these old periods. The `endAccShare` is only updated via the `__reset()` function, which is exclusively triggered by `setNewPeriod()`. This design means that stakers must wait for a new period to start before they can claim rewards from an expired period, introducing potential delays.

Recommendation

Consider adding a function to manually update `endAccShare` once a period ends. Alternatively, include automatic update logic into the `claimOldRewards()` function, allowing anyone to refresh it upon expiration.

Alleviation

[Seedify Finance Team, 10/24/2023]: The team heeded the advice and added the `__saveOldPeriod()` function into `claimOldRewards()` in commit 49e987095e999cee9602ea9421452162ef57a354. This function is designed to enable users to update expired periods.

[CertiK]: However, the introduction of the `__saveOldPeriod()` function led to an unintended issue: it mistakenly treated ongoing periods as expired, thereby incorrectly updating the `endAccShare` mapping. One variable affected was `accShare`, which determines stakers' rewards. Its value could be lower than it should be at the period's expiry, resulting in stakers claiming fewer rewards than anticipated after the period expires.

To address this, the team implemented a solution by adding a `block.timestamp > endingDate` check, as detailed in commit 626d304d632b06ab8b660f83ceeab33b916b50a7.

LFL-01 | THIRD-PARTY DEPENDENCY USAGE

Category	Severity	Location	Status
Design Issue	Minor	contracts/LockedFarming.sol (pre): 23, 25	Acknowledged

Description

The contract is serving as the underlying entity to interact with tokens `tokenAddress` and `rewardTokenAddress`. 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.

Recommendation

The auditors understood that the business logic requires interaction with third parties. It is recommended for the team to constantly monitor the statuses of third parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[Seedify Finance Team, 10/24/2023]: Issue acknowledged. I won't make any changes for the current version.

LFL-02 | POTENTIALLY LIMITS RECOVERY OF EXCESS TOKENS

Category	Severity	Location	Status
Logical Issue	Minor	contracts/LockedFarming.sol (pre): 686	Resolved

Description

The `recoverLostERC20()` function is designed to enable the `_owner` to retrieve tokens mistakenly sent to the contract. A safeguard mechanism is embedded within the function to ensure that the `_owner` cannot withdraw tokens (`tokenAddress`) that are actively staked by users. It achieves this by subtracting `stakedTotal` from the `amount` before the withdrawal.

However, there's a flaw in the logic: the `stakedTotal` represents the total amount of `tokenAddress` staked in the contract over its whole existence. Since it never decreases even when users make withdrawals, it could be larger the actual current staked balance. This flawed design means that the `_owner` might not be able to recover all the excess tokens beyond the current user stakes.

```
677     function recoverLostERC20(address token, address to) external onlyOwner {
678         if (token == address(0)) revert("Token_Zero_Address");
679         if (to == address(0)) revert("To_Zero_Address");
680 
681         uint256 amount = IERC20(token).balanceOf(address(this));
682 
683         // only retrieve lost {rewardTokenAddress}
684         if (token == rewardTokenAddress) amount -= rewardBalance;
685         // only retrieve lost LP tokens
686         if (token == tokenAddress) amount -= stakedTotal;
687 
688         IERC20(token).safeTransfer(to, amount);
689     }
```

Recommendation

Consider implementing a `currentStakedBalance` variable that increases upon deposits and decreases upon withdrawals. Use this value in place of `stakedTotal` within the `recoverLostERC20()` function to safeguard active stakes while allowing full recovery of any excess tokens.

Alleviation

[Seedify Finance Team, 10/24/2023]: The team heeded the advice and resolved the issue in commit 49e987095e999cee9602ea9421452162ef57a354.

LFL-03 INCOMPLETE INFORMATION RETRIEVAL FROM `userDeposits()`

Category	Severity	Location	Status
Design Issue	Informational	contracts/LockedFarming.sol (pre): 351	Resolved

Description

The `userDeposits()` function allows users to retrieve deposit details of a given address `from`. The `Deposits` struct contains a field name `userAccShare`, which represents the amount of rewards per wei of deposited LP token {tokenAddress} accross all periods. Notably, the `userDeposits()` function does not provide access to view the `userAccShare`, which appears to be an oversight.

```
351     /// @notice get user deposit details
352     function userDeposits(
353         address from
354     ) external view returns (uint256, uint256, uint256, uint256) {
355         if (hasStaked[from]) {
356             return (
357                 deposits[from].amount,
358                 deposits[from].initialStake,
359                 deposits[from].latestClaim,
360                 deposits[from].currentPeriod
361             );
362         } else {
363             return (0, 0, 0, 0);
364         }
365     }
```

```
84      struct Deposits {
85          uint256 amount;
86          uint256 initialStake;
87          uint256 latestClaim;
88          uint256 userAccShare;
89          uint256 currentPeriod;
90      }
```

Recommendation

It is recommended to add `deposits[from].userAccShare` to the return statement of the function.

Alleviation

[Seedify Finance Team, 10/24/2023]: The team heeded the advice and resolved the issue in commit 49e987095e999cee9602ea9421452162ef57a354.

LFL-04 FLAWED `stakedBalance` UPDATE MECHANISM IN `__withdraw()` FUNCTION

Category	Severity	Location	Status
Logical Issue	Informational	contracts/LockedFarming.sol (pre): 552	Resolved

Description

The `__withdraw()` function only decreases `stakedBalance` when `isPause` is false. The `stakedBalance` represents the amount of `tokenAddress` staked in the contract for the current period, and it directly impacts the reward calculations. Therefor, It should be updated whether the contract is paused or not. Failing to do so could lead to skewed reward distributions.

Specifically, the following scenario highlights the issue:

1. With `isPaused` being false, a user could invoke `emergencyWithdraw()` and reduce the actual staked balance without affecting the recorded `stakedBalance` in the contract.
2. When `isPaused` is set to true and `__updateShare()` function is triggered. When calculating the `accShare` (amount of rewards per wei), the formula uses the potentially inflated `stakedBalance` value, resulting in a smaller `accShare` than should be the case. This miscalculation means users might receive fewer rewards than they should.

It's worth mentioning that, in the current design, `isPaused` remains true after the setting of the first period, thus avoiding the described situation. Nevertheless, the logic appears faulty and could lead to unintended consequences if modified in the future.

Recommendation

It is recommended to revise the `__withdraw()` function, ensuring the `stakedBalance` is updated consistently if other requirements have been satisfied.

Alleviation

[Seedify Finance Team, 10/24/2023]: The team heeded the advice and resolved the issue in commit 49e987095e999cee9602ea9421452162ef57a354.

APPENDIX | SEEDIFY FINANCE LOCKED-YIELD-FARMING

Finding Categories

Categories	Description
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.
Design Issue	Design Issue findings indicate general issues at the design level beyond program logic that are not covered by other finding categories.

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 | CERTIK

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.

CertiK | Securing the Web3 World

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.