Creder (Stan Token) - audit

Security Assessment

CertiK Assessed on Mar 5th, 2025

CertiK Assessed on Mar 5th, 2025

Creder (Stan Token) - audit

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
ERC-20	EVM Compatible	Manual Review, Static Analysis
LANGUAGE	**TIMELINE**	**KEY COMPONENTS**
Solidity	Delivered on 03/05/2025	N/A

CODEBASE

Stan Token

View All in Codebase Page

COMMITS

Initial Commit: 22093f7c34ea7de92d8445c4fd281ffab18ad55b

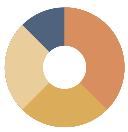
Remediation Commit: 1accd70994ecfe50b91fa409f1a5d04be28c6152

View All in Codebase Page

Highlighted Centralization Risks

- Privileged role can remove users' tokens
- Initial owner token share is 100%
- Has blacklist/whitelist

Vulnerability Summary



8	5	2	0	1	0
Total Findings	Resolved	Mitigated	Partially Resolved	Acknowledged	Declined

Count	Severity	Status	Description
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.
3	Major	2 Mitigated, 1 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.
2	Medium	2 Resolved	Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
2	Minor	2 Resolved	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.
1	Informational	1 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 | CREDER (STAN TOKEN) - AUDIT

CODEBASE | CREDER (STAN TOKEN) - AUDIT

Repository

Stan Token

Commit

Initial Commit: 22093f7c34ea7de92d8445c4fd281ffab18ad55b

Remediation Commit: 1accd70994ecfe50b91fa409f1a5d04be28c6152

AUDIT SCOPE | CREDER (STAN TOKEN) - AUDIT

1 file audited ● 1 file with Acknowledged findings

ID	Repo	File	SHA256 Checksum
● STS	CrederLabs/StanToken	contracts/StanToken.sol	080821f55ad2ad962a794762d5e15e5cc257975fb25a5e33378f815d9f46ed07

APPROACH & METHODS | CREDER (STAN TOKEN) - AUDIT

This report has been prepared for Creder to discover issues and vulnerabilities in the source code of the Creder (Stan Token) - audit 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.

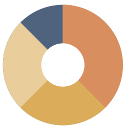
The auditing process pays special attention to the following considerations:

- 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 | CREDER (STAN TOKEN) - AUDIT



8	0	3	2	2	1
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for Creder (Stan Token) - audit. Through this audit, we have uncovered 8 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
STS-01	**Initial Token Distribution**	**Centralization**	**Major**	**Acknowledged**
STS-02	**Centralization Related Risks**	**Centralization**	**Major**	**Mitigated**
STS-03	**Owner's Ability To Arbitrarily Cancel Vestings**	**Centralization**	**Major**	**Mitigated**
STS-04	Variable `cancelHistory` Not Updated In `cancelLock`	Coding Issue	Medium	Resolved
STS-05	Incorrect Implementation Of The Transfer Function	Logical Issue	Medium	Resolved
STS-06	Unchecked ERC-20 `transfer()` / `transferFrom()` Call	Volatile Code	Minor	Resolved
STS-07	Ineffective Custom Reentrancy Guard	Logical Issue	Minor	Resolved
STS-08	Lack Of Zero Balance Validation In `cancelLock` Function	Logical Issue	Informational	Resolved

STS-01 | INITIAL TOKEN DISTRIBUTION

Category	Severity	Location	Status
Centralization	● **Major**	**contracts/StanToken.sol: 10~11**	● **Acknowledged**

Description

All of the `STAN` tokens are sent to the contract deployer on deployment. This is a centralization risk because the deployer can distribute tokens without obtaining the consensus of the community. Any compromise to these addresses may allow a hacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It is recommended that the team be transparent regarding the initial token distribution process. The token distribution plan should be published in a public location that the community can access. The team should make efforts to restrict access to the private keys of the deployer account or EOAs. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to a private key compromise.

Alleviation

[StanToken Team, 10/30/2024]: Issue acknowledged. I won't make any changes for the current version.

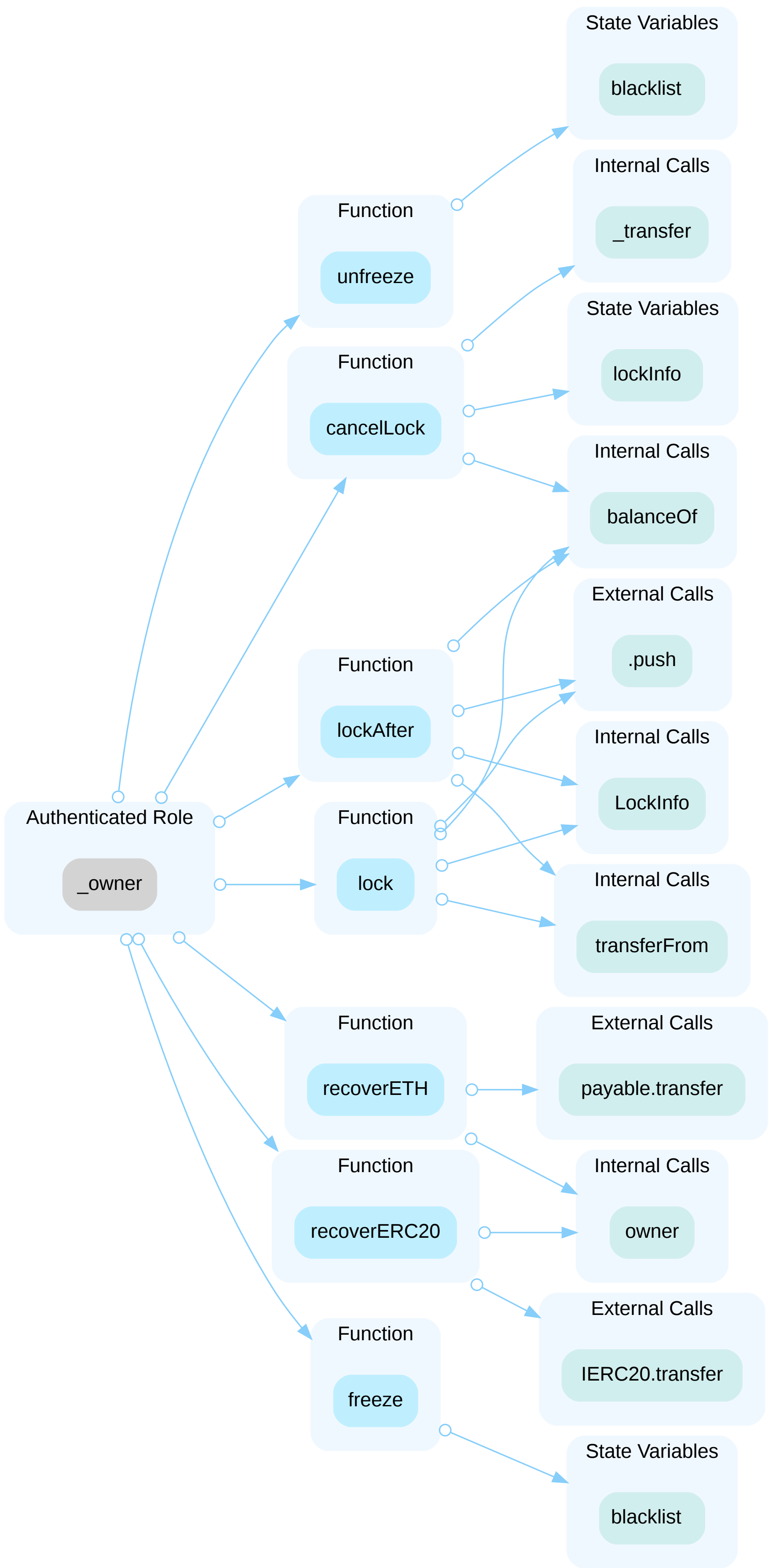
Given the current structure of StanToken's deployment, we have determined that publishing an initial distribution plan and implementing a multi-signature wallet is not essential. Our project's nature requires flexibility in adjusting distribution plans through further community consensus following the initial release. Token distribution will be limited to specific wallets with a well-defined security process to ensure optimal safety in token management.

STS-02 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	**Major**	**contracts/StanToken.sol: 30, 36, 113, 246, 257, 270, 295, 299**	**Mitigated**

Description

In the contract `StanToken` , 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 unfreeze the specified address, cancel lock and transfer balance back to owner, lock tokens for recipient after specified time, lock tokens until future release time, recover eth to owner, recover ERC20 tokens, and freeze an address by adding to blacklist.


State Variables
blacklist
Internal Calls
_transfer
Function
unfreeze
State Variables
lockInfo
Function
cancelLock
Internal Calls
balanceOf
External Calls
.push
Function
lockAfter
Internal Calls
LockInfo
Authenticated Role
_owner
Function
lock
Internal Calls
transferFrom
Function
recoverETH
External Calls
payable.transfer
Function
recoverERC20
Internal Calls
owner
External Calls
IERC20.transfer
Function
freeze
State Variables
blacklist

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

[StanToken Team, 10/30/2024]: Issue acknowledged.

In response to Certik's recommendations on enhancing access control and decentralization for sensitive functions, we have implemented three distinct roles to manage specific contract functionalities: BLACKLIST_MANAGER_ROLE: This role has exclusive access to manage the blacklist functionality (freeze and unfreeze), ensuring only designated addresses can control this feature. LOCK_MANAGER_ROLE: This role is responsible for the token locking and unlocking functions (lock, lockAfter, and cancelLock). By separating this role, we maintain stricter control over the locking operations, enabling more secure management of token vesting. RECOVERY_MANAGER_ROLE: This role is assigned to handle asset recovery functions (recoverERC20 and recoverETH). Only accounts with this role can recover tokens or ETH from the contract, further decentralizing authority over contract funds. With these roles, the contract restricts access to each function based on specific permissions, which are assigned by the owner through the DEFAULT_ADMIN_ROLE. This setup enables controlled access to critical functions and mitigates the risks associated with a single centralized account by decentralizing privileges across designated roles.

[StanToken Team, 11/06/2024]: The team acknowledged the issue and adopted the multisign solution to ensure the privileged role management process at the current stage. Changes have been made in this commit:

https://github.com/CrederLabs/StanToken/commit/898a37d18aeeb325ae4d6cfea383a06dda4c8e65

[CertiK, 11/06/2024]: While this strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team periodically revisit the private key security management of all signer addresses.

STS-03 | OWNER'S ABILITY TO ARBITRARILY CANCEL VESTINGS

Category	Severity	Location	Status
Centralization	Major	contracts/StanToken.sol: 270~283	Mitigated

Description

In the `StanToken` smart contract, the owner has the authority to cancel any user's token lock and reclaim the tokens through the `cancelLock` function:

```
    function cancelLock(address _holder, uint256 i) public onlyOwner {
      require(i < lockInfo[_holder].length, "No lock information.");

      uint256 amount = lockInfo[_holder][i].balance;

      require(super.balanceOf(address(this)) >= amount, "STAN Balance is too
small.");

      lockInfo[_holder][i].balance = 0;

      // The canceled amount is transferred back to the owner (since it has already
been transferred to this contract).
      _transfer(address(this), msg.sender, amount);

      emit CancelLock(_holder, amount);
  }
```

The owner can unilaterally cancel any user's token lock without restrictions or conditions. There is no requirement for user consent or notification prior to cancellation.

This would introduce centralization concerns. Token holders may lose confidence in the case when their vested tokens can be canceled and reclaimed by the owner without any restrictions. This undermines the perceived security and reliability of the token's vesting mechanism.

Besides, the owner can call `recoverERC20()` to transfer any ERC20 tokens from the `StanToken` contract, including the `StanToken` ifself:

```
    function recoverERC20(address tokenAddress, uint256 tokenAmount) public
onlyOwner {
        IERC20(tokenAddress).transfer(owner(), tokenAmount);
    }
```

If the owner withdraws a portion of the tokens before the release end time, user call to the `release()` function may fail due to insufficient balance.

Recommendation

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.

Alleviation

[StanToken Team, 11/06/2024]: The team acknowledged the issue and adopted the multisign solution to ensure the privileged role management process at the current stage. Changes have been made in this commit:

https://github.com/CrederLabs/StanToken/commit/898a37d18aeeb325ae4d6cfea383a06dda4c8e65

[CertiK, 11/06/2024]: While this strategy has indeed reduced the risk, it's crucial to note that it has not completely eliminated it. CertiK strongly encourages the project team periodically revisit the private key security management of all signer addresses.

STS-04 | VARIABLE `cancelHistory` NOT UPDATED IN `cancelLock`

Category	Severity	Location	Status
Coding Issue	Medium	contracts/StanToken.sol: 79, 201~203, 205~207, 209~211, 213~223	Resolved

Description

Variable `cancelHistory` is used without being initialized or updated.

In the `StanToken` contract, the `cancelHistory` mapping is intended to keep track of the history of canceled token locks for each address:

```
79      mapping(address => CancelHistory[]) internal cancelHistory;
```

However, the `cancelHistory` mapping is not updated within the `cancelLock` function where cancellations occur. As a result, when functions that retrieve cancellation history are called, they return empty or outdated data:

```
- `StanToken.cancelHistoryCount()`
- `StanToken.cancelHistoryState()`
- `StanToken.cancelHistoryStates()`
- `StanToken.cancelHistoryStates2()`
```

Recommendation

To maintain accurate records of canceled locks, we recommend modify the `cancelLock` function to record the cancellation details in the `cancelHistory` mapping.

Alleviation

[StanToken Team, 10/30/2024]: The team heeded the advice and resolved the issue in this commit.

STS-05 | INCORRECT IMPLEMENTATION OF THE TRANSFER FUNCTION

Category	Severity	Location	Status
Logical Issue	Medium	contracts/StanToken.sol: 47~51	Resolved

Description

The transfer function is incorrectly implemented as an internal function with the wrong signature:

```
    function transfer(address sender, address recipient, uint256 amount) internal
virtual whenNotPaused returns (bool) {
      require(!blacklist[sender] && !blacklist[recipient], "The user is frozen");
      super._transfer(sender, recipient, amount);
      return true;
  }
```

This function is internal and does not match the ERC20 standard transfer function signature, which is:

```
    function transfer(address recipient, uint256 amount) public virtual override
returns (bool)
```

As a result, external calls to transfer will invoke the inherited ERC20 transfer function, which lacks the blacklisting and pausing checks.

Recommendation

It's recommended to override the Internal `_transfer` function instead of the `transfer` function.

Alleviation

[StanToken Team, 10/30/2024]: The team heeded the advice and resolved the issue in commit: https://github.com/CrederLabs/StanToken/commit/a453aa58204a8a0e4ecf4c52967b208c44f3168d

STS-06 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	Minor	contracts/StanToken.sol: 295~297	Resolved

Description

The return values of the `transfer()` and `transferFrom()` calls in the smart contract are not checked. Some ERC-20 tokens' transfer functions return no values, while others return a bool value, they should be handled with care. If a function returns `false` instead of reverting upon failure, an unchecked failed transfer could be mistakenly considered successful in the contract.

```
296         IERC20(tokenAddress).transfer(owner(), tokenAmount);
```

Recommendation

It is advised to use the OpenZeppelin's `SafeERC20.sol` implementation to interact with the `transfer()` and `transferFrom()` functions of external ERC-20 tokens. The OpenZeppelin implementation checks for the existence of a return value and reverts if false is returned, making it compatible with all ERC-20 token implementations.

Alleviation

[StanToken Team, 10/30/2024]: The team heeded the advice and resolved the issue in commit: https://github.com/CrederLabs/StanToken/commit/fdfe4fdd64a48ca80bc0e4a69f54f694c0fa4f5b

STS-07 | INEFFECTIVE CUSTOM REENTRANCY GUARD

Category	Severity	Location	Status
Logical Issue	Minor	contracts/StanToken.sol: 17~25	Resolved

Description

The custom reentrancy guard uses a mapping with msg.sender as the key:

```
    mapping (address => bool) private _locks;

    modifier nonReentrant {
      require(_locks[msg.sender] != true, "ReentrancyGuard: reentrant call");
      _locks[msg.sender] = true;
      _;
      _locks[msg.sender] = false;
}
```

If a reentrant call changes msg.sender (e.g., via a proxy contract), the guard may not prevent reentrancy.

Recommendation

It's recommended to implement OpenZeppelin's ReentrancyGuard, which uses a single `status` variable.

```
    modifier nonReentrant {
      require(_status != _ENTERED, "ReentrancyGuard: reentrant call");
      _status = _ENTERED;
      _;
      _status = _NOT_ENTERED;
  }
```

Alleviation

[StanToken Team, 10/30/2024]: The team heeded the advice and resolved the issue in commit: https://github.com/CrederLabs/StanToken/commit/6d229abee757fca1a5d3e10f9700af480f380bc8

STS-08 | LACK OF ZERO BALANCE VALIDATION IN `cancelLock` FUNCTION

Category	Severity	Location	Status
Logical Issue	● Informational	contracts/StanToken.sol: 270~283	● Resolved

Description

The `cancelLock` function does not check if the lock has already been released or canceled (i.e., if the balance is zero).

```
270     function cancelLock(address _holder, uint256 i) public onlyOwner {
271         require(i < lockInfo[_holder].length, "No lock information.");
272
273         uint256 amount = lockInfo[_holder][i].balance;
274
275         require(super.balanceOf(address(this)) >= amount,
"STAN Balance is too small.");
276
277         lockInfo[_holder][i].balance = 0;
278
279
// The canceled amount is transferred back to the owner (since it has already been
transferred to this contract).

280         _transfer(address(this), msg.sender, amount);
281
282         emit CancelLock(_holder, amount);
283     }
```

Attempting to cancel an already released or canceled lock is unnecessary and will waste gas.

Recommendation

It's recommended to add validation checks on `lockInfo` balance.

Alleviation

[StanToken Team, 10/30/2024]: The team heeded the advice and resolved the issue in commit: https://github.com/CrederLabs/StanToken/commit/e14e5897d63065f4ecf6f6ad9196452c3930f586

APPENDIX | CREDER (STAN TOKEN) - AUDIT

Finding Categories

Categories	Description
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
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.

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.

Elevating Your Entire Web3 Journey

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.