CERTIK

Security Assessment

HanIdentity - Addendum

CertiK Assessed on Jun 27th, 2023

CertiK Assessed on Jun 27th, 2023

HanIdentity - Addendum

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

Executive Summary

TYPES	ECOSYSTEM	METHODS
ERC-20	Ethereum (ETH)	Manual Review, Static Analysis

LANGUAGE	TIMELINE	KEY COMPONENTS
Solidity	Delivered on 06/27/2023	N/A

CODEBASE

https://github.com/hanchain-paykhan/staking.paykhan.io/tree/4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81/contract/governance

View All in Codebase Page

COMMITS

4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81

081e5389ef5e8a1c7769414899927931727bcf10

aba36a9082f4d3bde15828a153ea37f2b9903947

View All in Codebase Page

Vulnerability Summary

11	8	1	0	2	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.
5	Major	3 Resolved, 1 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.
0	Medium		Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.
4	Minor	4 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.
2	Informational	1 Resolved, 1 Acknowledged	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 | HANIDENTITY - ADDENDUM

CODEBASE | HANIDENTITY - ADDENDUM

Repository

https://github.com/hanchain-paykhan/staking.paykhan.io/tree/4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81/contract/governance

https://github.com/hanchain-paykhan/hanchain/tree/081e5389ef5e8a1c7769414899927931727bcf10/contracts/TokenVestingLock.sol

https://github.com/hanchain-paykhan/HANePlatform/blob/aba36a9082f4d3bde15828a153ea37f2b9903947/contracts/HANePlatform.sol

Commit

4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81
081e5389ef5e8a1c7769414899927931727bcf10
aba36a9082f4d3bde15828a153ea37f2b9903947

AUDIT SCOPE | HANIDENTITY - ADDENDUM

4 files audited ● 1 file with Acknowledged findings ● 1 file with Resolved findings ● 2 files without findings

ID	Repo	File	SHA256 Checksum
● HGB	hanchain-paykhan/staking.paykhan.io	HanGovernor.sol	899e4b0a194b702e007d8ed3f53d5006889df5478b1632ae90a91ae13abac719
● HTC	hanchain-paykhan/staking.paykhan.io	HanTimelockController.sol	65520c4423ff2ada66696447839a1d8e11c7d9de0103489099424ed0de5d011a
● TVL	hanchain-paykhan/hanchain	TokenVestingLock.sol	743da16d98b6caedb2c11b9e3e33ec6abbc7723f05f5bc7d56945947a458ed78
● HAP	hanchain-paykhan/HANePlatform	HANePlatform.sol	9a341cb7c4f00d0a93f148a7aa46940348324d9049347eec0ee727b4410b9a33

APPROACH & METHODS | HANIDENTITY - ADDENDUM

This report has been prepared for HanIdentity to discover issues and vulnerabilities in the source code of the HanIdentity - Addendum 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.

REVIEW NOTES | HANIDENTITY - ADDENDUM

Overview

HanIdentity has implemented several Solidity smart contracts that defines their own `Governor`, `HANePlatform Token`, `TimeLockController` and `TokenVestingLock`, utilizing OpenZeppelin library contracts(Not within the scope of the audit, assuming it is OpenZeppelin (oz)).

The contracts inherits from OpenZeppelin contracts:

- @openzeppelin/contracts/governance/Governor.sol
- @openzeppelin/contracts/governance/extensions/GovernorSettings.sol
- @openzeppelin/contracts/governance/extensions/GovernorCountingSimple.sol
- @openzeppelin/contracts/governance/extensions/GovernorVotes.sol
- @openzeppelin/contracts/governance/extensions/GovernorVotesQuorumFraction.sol
- @openzeppelin/contracts/governance/extensions/GovernorTimelockControl.sol
- @openzeppelin/contracts/governance/TimelockController.sol
- @openzeppelin/contracts/token/ERC20/IERC20.sol
- @openzeppelin/contracts/token/ERC721/IERC721.sol
- @openzeppelin/contracts/access/Ownable.sol
- @openzeppelin/contracts/token/ERC20/ERC20.sol
- @openzeppelin/contracts/token/ERC20/extensions/ERC20Burnable.sol

FINDINGS | HANIDENTITY - ADDENDUM

11	0	5	0	4	2
Total Findings	Critical	Major	Medium	Minor	Informational

This report has been prepared to discover issues and vulnerabilities for HanIdentity - Addendum. Through this audit, we have uncovered 11 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
HAN-01	**Centralization Related Risks**	**Centralization**	**Major**	**Resolved**
HAP-01	**Initial Distribution Centralization Risk**	**Centralization**	**Major**	**Mitigated**
HGB-01	**Centralization Risks In HanGovernor.Sol**	**Centralization, Governance**	**Major**	**Acknowledged**
HTC-01	**Unnecessary Centralized Functions**	**Centralization**	**Major**	**Resolved**
HTC-02	**Centralization Risks In HanTimelockController.Sol**	**Centralization, Governance**	**Major**	**Resolved**
TVL-01	Meaningless Comparison	Logical Issue	Minor	Resolved
TVL-02	Division Before Multiplication	Incorrect Calculation	Minor	Resolved
TVL-04	Missing Duplicate Check	Logical Issue, Inconsistency	Minor	Resolved
TVL-05	Lack Of Input Validation	Volatile Code	Minor	Resolved
TVL-06	Incorrect Event Parameter	Logical Issue	Informational	Resolved
TVL-08	Lack Of Balance Check	Code Optimization	Informational	Acknowledged

HAN-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization	Major	TokenVestingLock.sol (081e5389ef5e8a1c7769414899927931727bcf10): 189, 205; HanTimelockController.sol (4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81): 16, 24	Resolved

Description

In the contract `HanTimelockController` the role `_owner` has authority over the functions shown below:

- recoverERC20(): withdraw any ERC20 tokens from the contract to the `owner` account.
- recoverERC721(): withdraw any ERC721 tokens from the contract to the `owner` account.

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

- Withdraw all ERC20 and ERC721 assets from the contract.

In the contract `TokenVestingLock` the role `_owner` has authority over the functions shown below:

- recoverERC20(): withdraw any ERC20 tokens from the contract to the `owner` account. If the token is the release token, the unreleased portion will be kept in the contract.
- recoverERC721(): withdraw any ERC721 tokens from the contract to the `owner` account.

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

- Withdraw ERC20 and ERC721 assets from the contract.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., 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

[CertiK]:

The team has removed the centralized functions mentioned above in the commit: 7f08dec6dda060ca45173e6d9fb1ab8377210d0b and 1d3cc1d263aff9158522c201cc80fc0970a0c2ad.

HAP-01 | INITIAL DISTRIBUTION CENTRALIZATION RISK

Category	Severity	Location	Status
Centralization	**Major**	**HANePlatform.sol (aba36a9082f4d3bde15828a153ea37f2b9903947): 9**	**Mitigated**

Description

All of the **HANeP** tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the anonymous deployer can distribute tokens without obtaining the consensus of the community. Any compromise to the deployer account that holds undistributed tokens may allow the attacker to steal and sell tokens on the market, resulting in severe damage to the project.

Recommendation

It's recommended 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 shall make enough efforts to restrict the access of the private key. A multi-signature (⅔, ⅗) wallet can be used to prevent a single point of failure due to the private key compromise. Additionally, the team can lock up a portion of tokens, release them with a vesting schedule for long-term success, and deanonymize project teams with a third-party KYC provider to create greater accountability.

Alleviation

[Hanidentity]: We are holding tokens issued on a multi-signature(2/3) wallet, which is regularly disclosed to the community. https://medium.com/@HANeP/haneplatform-tokenomics-514d6dc5e478.

These wallets are meticulously managed in an isolated cold wallet room, where they can only operate as hardware wallets according to standardized security protocols. Once the `TokenVestingLock` contract under audit is completed, the tokens will be locked and released according to the vesting schedule.

[CertiK]: The team has applied ⅔ multisig solution as their short-term solution. Below is the configuration information.
Multi-signature wallet addresses:
MutiSigWallet_ePlatform (IAO):
0x495fcd7f56a0bf8be1f29be02d1aa5f492f2ff66

MutiSigWallet_Partner:
0x19681f34afce6b7fadfb07cd34c8f20dcf0a4f2a

MutiSigWallet_Founder:
0x90a692e0819075c49100f9f5f2724e75d8a34711

MutiSigWallet_Team and Advisor:
0xc7bdbcda0b8162427868ac41713d2559a9e2281c

MutiSigWallet_Reward (Staking):
0x3811f5674abbc216ad29a1edcdd0b05172a9f123

Each wallet has three owners:
1.0xfDB509381b0dEdde0599607aFd92C935CAdC3Ef7
2.0x60A3fc3f8E68C3561d52697cD14f9C0c4fBa4b9A
3.0xA137120BCC903638CF156c6F66b5c24997630722

Threshold is 2.

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 above-listed addresses.

HGB-01 | CENTRALIZATION RISKS IN HANGOVERNOR.SOL

Category	Severity	Location	Status
Centralization, Governance	**Major**	**HanGovernor.sol (4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81): 27**	**Acknowledged**

Description

Contract `HanGovernor` inherits from some `openzeppelin` contracts which are related to governance design. One important point is that if a time lock was set, then the time lock could play as a governance role and execute some privileged functions. However, if the time lock wasn't set correctly, centralization issues may occur.

```
constructor(
    IVotes _token,
    TimelockController _timelock
)
    Governor("HanGovernor")
    GovernorSettings(1 /* 1 block */, 50400 /* 1 week */, 0)
    GovernorVotes(_token)
    GovernorVotesQuorumFraction(4)
    GovernorTimelockControl(_timelock)
{}
```

Privileged functions:

- relay(): Relays a transaction or function call to an arbitrary target.
- setVotingDelay(): Update the voting delay.
- setVotingPeriod(): Update the voting period.
- setProposalThreshold(): Update the proposal threshold.
- updateQuorumNumerator(): Changes the quorum numerator.
- updateTimelock(): Public endpoint to update the underlying timelock instance.

Any compromise to the time lock address may allow the hacker to take advantage of this authority and gain benefits by influencing governance functions.

Recommendation

We recommend checking if the time lock contract address is set correctly before using governance function.

Alleviation

[HanIdentity]: Acknowledged.

HTC-01 | UNNECESSARY CENTRALIZED FUNCTIONS

Category	Severity	Location	Status
Centralization	**Major**	**HanTimelockController.sol (4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81): 16, 24**	**Resolved**

Description

In the contract `HanTimelockController` the role `_owner` has authority over the functions shown below:

- recoverERC20()
- recoverERC721()

Assuming a proposal requires the contract `TimeLockController` to act as a temporary fund intermediary and the proposal is approved. At some point, the contract `TimeLockController` may hold a significant amount of funds.

If the owner's account private key is compromised, a hacker could withdraw all ERC20 and ERC721 assets from the contract, resulting in substantial losses.

From a design perspective, since the contract does not have a need to withdraw funds, we consider the mentioned centralized functions to be irrelevant.

Recommendation

We recommend removing these unnecessary centralized functions.

Alleviation

[CertiK]:

The team has removed the centralized functions mentioned above in the commit:7f08dec6dda060ca45173e6d9fb1ab8377210d0b.

HTC-02 CENTRALIZATION RISKS IN HANTIMELOCKCONTROLLER.SOL

Category	Severity	Location	Status
Centralization, Governance	**Major**	**HanTimelockController.sol (4fcd614e0bd4bff1911d2be8a9bec5fa5575bf81): 14**	**Resolved**

Description

Based on the linked code, the deployer of the contract `HanTimelockController` will be granted the admin role. With this role, the deployer could influence the time lock deeply. This cause the time lock to lose its meaning. An optional admin can assist with the initial configuration of the role after deployment without delay, but this role should then be dropped to support administration through time locked proposals.

```
constructor(
     uint256 minDelay,
     address[] memory proposers,
     address[] memory executors
 ) TimelockController(minDelay, proposers, executors, msg.sender) {}
```

Recommendation

We recommend renouncing the admin role before using the time lock controller functionality.

Alleviation

[CertiK]:

The team has changed the relevant code so that the contract no longer sets the deployer as the `DEFAULT_ADMIN_ROLE` during deployment. Changes have been reflected in the commit: ac98f62bac53f5771420f9ec690607adf21c0654.

TVL-01 | MEANINGLESS COMPARISON

Category	Severity	Location	Status
Logical Issue	Minor	TokenVestingLock.sol (081e5389ef5e8a1c7769414899927931727bcf10): 102	Resolved

Description

According to the formula for the local variable `nextIntervalTime`, its value is a time period, most likely less than the current time. Therefore, the linked `if` judgement will always return true and is therefore meaningless.

```
1    uint256 nextIntervalTime = (currentInterval + 1) * intervalSeconds;
2
3    if (nextIntervalTime <= currentTime) {
```

Recommendation

We recommend removing the `if` judgement if unnecessary.

Alleviation

[CertiK]:

The team heeded the advice and resolved the finding in the commit: 43ac5320c116c8aa02ed6816007eb4db2b5de3a8.

TVL-02 | DIVISION BEFORE MULTIPLICATION

Category	Severity	Location	Status
Incorrect Calculation	Minor	TokenVestingLock.sol (081e5389ef5e8a1c776941489992793172 7bcf10): 81	Resolved

Description

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

```
81  uint256 tokensPerRoundPerBeneficiary = totalReleaseTokens * intervalSeconds /
 durationSeconds * _shares[i] / 100;
```

Recommendation

We recommend applying multiplications before divisions if integer overflow does not happen in functions.

Alleviation

[CertiK]:

The team heeded the advice and resolved the finding in the commit: 358053ca9143bbce683b8c0e0985461c7af8b1b0.

TVL-04 | MISSING DUPLICATE CHECK

Category	Severity	Location	Status
Logical Issue, Inconsistency	Minor	TokenVestingLock.sol (081e5389ef5e8a1c77694148999279 31727bcf10): 63	Resolved

Description

According to the comments above the function `constructor()`, the `accounts` cannot contains duplicate account.

```
1
/* All addresses in `accounts` must be non-zero. Both arrays must have the same non-
zero length, and there must be no
2       * duplicates in `accounts`
3     */
```

Recommendation

We recommend adding a duplicate check for the input value `_accounts`.

Alleviation

[CertiK]:

The team heeded the advice and resolved the finding in the commit: 1d3cc1d263aff9158522c201cc80fc0970a0c2ad.

TVL-05 | LACK OF INPUT VALIDATION

Category	Severity	Location	Status
Volatile Code	Minor	TokenVestingLock.sol (081e5389ef5e8a1c7769414899927931727bcf10): 81	Resolved

Description

According to the design, `durationSeconds` represents the duration of the vesting period in seconds, `intervalSeconds` represents the duration of each round in seconds. The user's earnings are based on the following calculation formula:

```
 81  uint256 tokensPerRoundPerBeneficiary = totalReleaseTokens * intervalSeconds /
durationSeconds * _shares[i] / 100;
 82  uint256 releaseTokens = tokensPerRoundPerBeneficiary * totalRounds;
```

However, the function does not verify whether `durationSeconds` can be evenly divided by `intervalSeconds`. If it cannot, the result calculated based on the above formula will not match the actual outcome.

Recommendation

We recommend adding a `require` check as below:

```
require(durationSeconds % intervalSeconds == 0, "error durationSeconds value");
```

Alleviation

[CertiK]:
The team heeded the advice and resolved the finding in the commit: f64246e22bb992629b8daee402a69e966caa68a9.

TVL-06 | INCORRECT EVENT PARAMETER

Category	Severity	Location	Status
Logical Issue	Informational	TokenVestingLock.sol (081e5389ef5e8a1c776941489992793172 7bcf10): 194	Resolved

Description

According to the logic of the function `recoverERC20()`, `_tokenAmount` tokens will be transferred to the owner. However, the value recorded in the event is the maximum token amount that can be recovered but not the real amount that was transferred.

```
IERC20(_tokenAddress).transfer(msg.sender, _tokenAmount);
emit RecoveredERC20(_tokenAddress, tokenAmount);
```

Recommendation

We recommend correcting the parameter passed to the event.

Alleviation

[CertiK]:

The team has removed the relevant code in the commit: 1d3cc1d263aff9158522c201cc80fc0970a0c2ad.

TVL-08 | LACK OF BALANCE CHECK

Category	Severity	Location	Status
Code Optimization	Informational	TokenVestingLock.sol (081e5389ef5e8a1c7769414899927931727bcf10): 120	Acknowledged

Description

The function `release()` is used to releases tokens to payees based on the vesting schedule. According to the logic of the contract `TokenVestingLock.sol`, token distributors will send tokens to this contract directly, then anyone can execute the function `release()` to release tokens to payees. Since tokens are sent by distributors, it is best to determine if the current balance is sufficient, considering that the number of tokens sent to this contract may not be sufficient.

```
1     token.transfer(payees[i].account, tokensToRelease);
```

Recommendation

We recommend adding logic that send available balance to payees. For example:

```
117     if(tokensToRelease > token.balanceOf(address(this))) tokensToRelease = token
.balanceOf(address(this));
118     releasedAmount[payees[i].account] += tokensToRelease;
119     unreleased -= tokensToRelease;
120     totalReleasedTokens += tokensToRelease;
121     token.transfer(payees[i].account, tokensToRelease);
```

Alleviation

[HanIdentity]:
Immediately after deployment, only the accurately calculated tokens are transferred to the deployed contract. If there was an error in the calculation and an incorrect quantity of tokens was transferred to the contract, the `release()` function will revert, and additional tokens will be transferred to the contract.

OPTIMIZATIONS | HANIDENTITY - ADDENDUM

ID	Title	Category	Severity	Status
TVL-03	Meaningless Variable	Logical Issue	Optimization	Resolved
TVL-07	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	Resolved

TVL-03 | MEANINGLESS VARIABLE

Category	Severity	Location	Status
Logical Issue	Optimization	TokenVestingLock.sol (081e5389ef5e8a1c776941489992793172 7bcf10): 110	Resolved

Description

In #L110, the local variable `unreleased` was assigned with amount of tokens released so far. The `for` loop below will release all tokens after the last release. Therefore, deducting `tokensToRelease` this time from `unreleased` is meaningless since `unreleased` contains the amount that had been released before. The value of `unreleased` is greater than 0 definitely. Even in the case that release all tokens at once, it is still meaningless because the sum of the released tokens is smaller than or equal to the whole amount will be released.

```
    uint256 unreleased = totalVestedTokens;
    for (uint256 i = 0; i < payees.length; i++) {
        uint256 payeeShare = (payees[i].shares * totalVestedTokens) / 100;
        uint256 releasable = payeeShare - releasedAmount[payees[i].account];

        if (unreleased > 0 && releasable > 0) {
            uint256 tokensToRelease = (releasable < unreleased) ? releasable :
unreleased;
            releasedAmount[payees[i].account] += tokensToRelease;
            unreleased -= tokensToRelease;
            totalReleasedTokens += tokensToRelease;
            token.transfer(payees[i].account, tokensToRelease);
            emit released(payees[i].account, tokensToRelease);
        }
    }
```

Recommendation

We recommend removing the meaningless variable and related logic.

Alleviation

[CertiK]:

The team heeded the advice and resolved the finding in the commit: f64246e22bb992629b8daee402a69e966caa68a9.

TVL-07 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	Optimization	TokenVestingLock.sol (081e5389ef5e8a1c7769414899927931727bcf10): 40~42, 44~46	Resolved

Description

The linked variables assigned in the constructor can be declared as `immutable`. Immutable state variables can be assigned during contract creation but will remain constant throughout the lifetime of a deployed contract. A big advantage of immutable variables is that reading them is significantly cheaper than reading from regular state variables since they will not be stored in storage.

Recommendation

We recommend declaring these variables as immutable. Please note that the `immutable` keyword only works in Solidity version `v0.6.5` and up.

Alleviation

[CertiK]:

The team heeded the advice and resolved the finding in the commit: 358053ca9143bbce683b8c0e0985461c7af8b1b0.

APPENDIX | HANIDENTITY - ADDENDUM

Finding Categories

Categories	Description
Centralization	Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.
Gas Optimization	Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.
Logical Issue	Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.
Inconsistency	Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.
Incorrect Calculation	Incorrect Calculation findings indicate incorrect caculation such as computation not according to the design, precision errors, rounding errors, etc.
Governance	Governance finding describe governance issue such as voting threshold being set too low, voting structure issue, etc.

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.