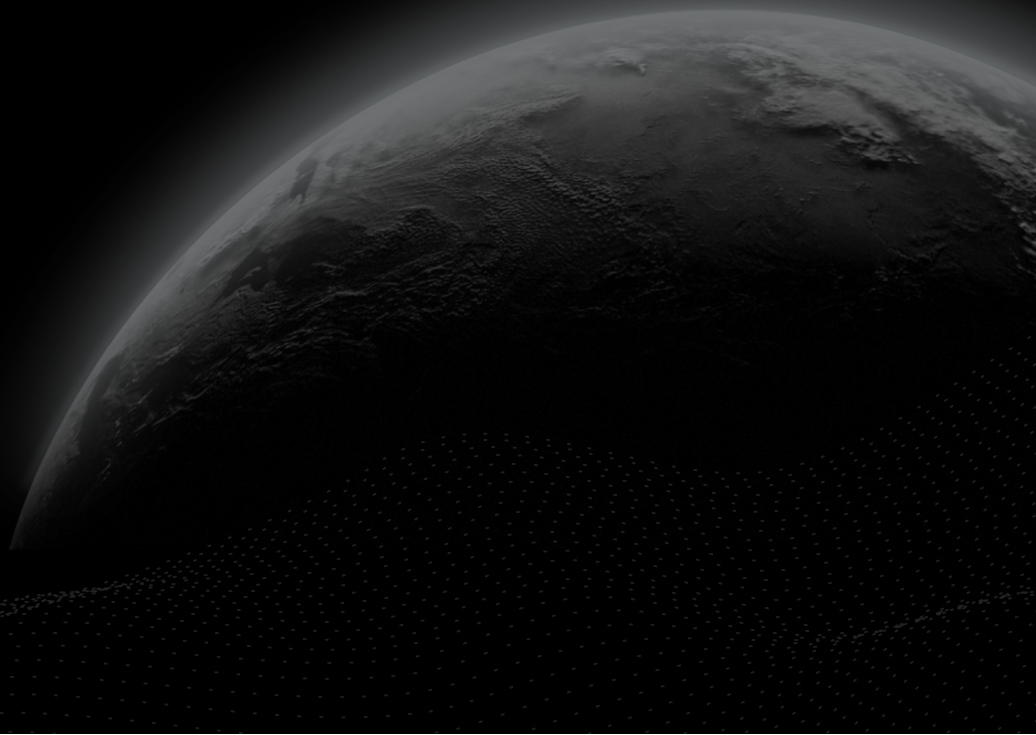




Security Assessment

Betfin High Low Contracts

CertiK Assessed on Sept 30th, 2024





CertiK Assessed on Sept 30th, 2024

Betfin High Low Contracts

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

Executive Summary

TYPES DeFi	ECOSYSTEM Ethereum (ETH)	METHODS Formal Verification, Manual Review, Static Analysis
LANGUAGE Solidity	TIMELINE Delivered on 09/30/2024	KEY COMPONENTS N/A
CODEBASE hilo-contract View All in Codebase Page	COMMITTS <ul style="list-style-type: none"> • Initial: 3c20e9f280625d0fe07b9e45c50b7b0ac5f3b747 • Updated1: ad183550821acc7887fb22aad19d9c8b2791969e • Updated2: 8177facfb1cf5cc7f9e5472c87e83939a38e72bc View All in Codebase Page	

Vulnerability Summary



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.
0 Major		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.
3 Medium	3 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 | BETFIN HIGH LOW CONTRACTS

| Summary

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

| Review Notes

[Overview](#)

[External Dependencies](#)

| Findings

[HLB-04 : Potential Depletion of LINK Funds of Subscription Account Due to Repeated Small Bets](#)

[HLT-02 : Enhanced Security Through Increased Block Confirmations in Chainlink VRF Requests on Polygon Network](#)

[HLU-01 : Compiler Error](#)

[HLB-01 : Unchecked ERC-20 `transfer\(\)`/`transferFrom\(\)` Call](#)

[HLB-03 : Discrepancy in Random Number Range and Betting Threshold in `fulfillRandomWords` Function](#)

[HLB-02 : Security Risk Due to Presence of Test Function `FulfillRandomWords\(\)`](#)

| Optimizations

[HIG-01 : Variables That Could Be Declared as Immutable](#)

[HLT-01 : State Variable Should Be Declared Constant](#)

| Appendix

| Disclaimer

CODEBASE | BETFIN HIGH LOW CONTRACTS

Repository

[hilo-contract](#)











Commit

- [Initial: 3c20e9f280625d0fe07b9e45c50b7b0ac5f3b747](#)
- [Updated1: ad183550821acc7887fb22aad19d9c8b2791969e](#)
- [Updated2: 8177facfb1cf5cc7f9e5472c87e83939a38e72bc](#)
- [Final: cea0df2e46eeb1a8da260efcf08c3a05ff88a39e](#)

AUDIT SCOPE | BETFIN HIGH LOW CONTRACTS

10 files audited ● 4 files with Resolved findings ● 6 files without findings



ID	Repo	File	SHA256 Checksum
● HLB	betfinio/hilo-contract	 src/HighLow.sol	ffe20416332ae73a5073682103171aaba651aa07cb455e962ebdf6051f3e5f6
● HIG	betfinio/hilo-contract	 src/HighLowBet.sol	9d99fe639db31642363144dcae20839b96330b0c54fd7d6c01305a62322abd0e
● HLU	betfinio/hilo-contract	 src/HighLow.sol	efe0e84bdcc585dbfe856cee1d51a4270cb2ce98d59b63f4b1ec48015fe24d7c
● HLT	betfinio/hilo-contract	 src/HighLow.sol	face8cf7ee69964834192e8cd65bd56b9ca32bc84380d212724c6db04a72ab05
● HIH	betfinio/hilo-contract	 src/HighLowBet.sol	f20344c277c6d82b4fa24beed851512e445e28640eb69c9c4f3cd0fe6b76ccf3
● HLH	betfinio/hilo-contract	 src/HighLow.sol	d1c500df8ab597265eecf6066426c59a47134d937f26f6ef3c45cb0d6c2ecd4b
● HIL	betfinio/hilo-contract	 src/HighLowBet.sol	f20344c277c6d82b4fa24beed851512e445e28640eb69c9c4f3cd0fe6b76ccf3
● HIO	betfinio/hilo-contract	 src/HighLowBet.sol	f20344c277c6d82b4fa24beed851512e445e28640eb69c9c4f3cd0fe6b76ccf3
● HLI	betfinio/hilo-contract	 src/HighLow.sol	e54c022d7f74ff8b2b444f0956385d1e7f265904e0d8dc3b6f6dbec2eca6c554
● HIW	betfinio/hilo-contract	 src/HighLowBet.sol	f20344c277c6d82b4fa24beed851512e445e28640eb69c9c4f3cd0fe6b76ccf3

APPROACH & METHODS | BETFIN HIGH LOW CONTRACTS

This report has been prepared for Betfin.io to discover issues and vulnerabilities in the source code of the Betfin High Low Contracts project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Formal Verification, 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 | BETFIN HIGH LOW CONTRACTS

Overview

The **Betfin.io** project facilitates a betting game where players place bets on two sides in each bet. Once a bet is placed, the VRF oracle service generates a random number to determine whether the player wins. The contract manages player bets, reserves funds for payouts, and ensures secure fund transfers.

External Dependencies

In **Betfin.io**, the project relies on a few external contracts or addresses to fulfill the needs of its business logic.

The following are third dependencies contracts used within the `Dice` and `DiceBet` contracts:

- `openzeppelin`: including `AccessControl`, `ReentrancyGuard`, `IERC20`, `SafeERC20` and `Ownable`;
- `chainlink`: including `VRFCoordinatorV2_5` and `VRFConsumerBaseV2Plus`.

It is assumed that these contracts or addresses are trusted and properly implemented within the entire project.

The team utilizes the subscription method of the Chainlink VRF service to generate random numbers. It is assumed that the `subscriptionId` in the project is always valid and maintains a sufficient balance to fund requests from consumer contracts.

FINDINGS | BETFIN HIGH LOW CONTRACTS



6

Total Findings

0

Critical

0

Major

3

Medium

2

Minor

1

Informational

This report has been prepared to discover issues and vulnerabilities for Betfin High Low Contracts. Through this audit, we have uncovered 6 issues ranging from different severity levels. Utilizing the techniques of Formal Verification, Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
HLB-04	Potential Depletion Of LINK Funds Of Subscription Account Due To Repeated Small Bets	Design Issue	Medium	● Resolved
HLT-02	Enhanced Security Through Increased Block Confirmations In Chainlink VRF Requests On Polygon Network	Design Issue	Medium	● Resolved
HLU-01	Compiler Error	Coding Issue	Medium	● Resolved
HLB-01	Unchecked ERC-20 <code>transfer()</code> / <code>transferFrom()</code> Call	Volatile Code	Minor	● Resolved
HLB-03	Discrepancy In Random Number Range And Betting Threshold In <code>fulfillRandomWords</code> Function	Logical Issue	Minor	● Resolved
HLB-02	Security Risk Due To Presence Of Test Function <code>FulfillRandomWords()</code>	Logical Issue	Informational	● Resolved

HLB-04 | POTENTIAL DEPLETION OF LINK FUNDS OF SUBSCRIPTION ACCOUNT DUE TO REPEATED SMALL BETS

Category	Severity	Location	Status
Design Issue	● Medium	src/HighLow.sol (09/06-3c20e9): 28-33	● Resolved

Description

The `HighLow` contract uses Chainlink VRF v2.5 to generate random numbers for its gambling game. Each request for a random number incurs a fee, payable in LINK tokens, from the contract's subscription account with Chainlink.

The game's payout structure can multiply the bet amount by up to 10,000 times, significantly leveraging the potential payouts.

```
function getPossibleWin(
    uint256 _threshold,
    bool _side,
    uint256 _amount
) public pure returns (uint256) {
    require(_threshold > 0 && _threshold < 10000, "D04");

    // Calculate multiplier according to the success percentage
    if (!_side) {
        return (10000 * _amount) / _threshold;
    } else {
        return (10000 * _amount) / (10000 - _threshold);
    }
}
```

A malicious actor could exploit this by deploying an attack contract that repeatedly places a large number of bets with very small amounts. Each bet triggers a VRF request, incurring a LINK fee deducted from the subscription account. By flooding the contract with numerous small bets, an attacker could rapidly deplete the LINK funds in the subscription account. If the subscription fees are not replenished within 24 hours after depletion, the pending random number requests will expire. This expiry blocks the resolution of bets, consequently locking the reserved funds from the `Staking` contract in the `HighLow` contract indefinitely.

The auditing team would like to confirm with the team whether the current implementation is intended.

Recommendation

The auditing team would like to confirm with the team whether the current implementation is intended.

Alleviation

[Betfin Team, 09/19/2024]:

Issue acknowledged. Changes have been reflected in the commit hash: <https://github.com/betfinio/hilo-contract/commit/0cb63c1dd392cc8addce029ee44dd03b8b18a655>.

[CertiK, 09/20/2024]:

It's noted that when `placeBet()` checks whether the token amount meets the `MIN_BET`, it uses `amount` (the value decoded from calldata) for comparison. However, based on the code context, `amount` lacks precision and represents only a numerical value. Therefore, we suggest using `_amount`, which includes precision, instead of `amount`.

```
164         (address player, uint256 amount, uint256 _threshold, bool _side) = abi
165             .decode(_data, (address, uint256, uint256, bool));
166         //revert if player is not the same
167         require(player == _player, "D02");
168         //revert if amount is not whole
169         require(amount * 10 ** 18 == _amount, "D03");
170
171         @> require(amount >= MIN_BET, "D08");
```

[Betfin Team, 09/20/2024]:

Issue acknowledged. The team resolved this issue in the commit hash [cccc083c5d19a632180275ecc889ed7f40d1cd09](https://github.com/betfinio/hilo-contract/commit/cccc083c5d19a632180275ecc889ed7f40d1cd09).

[CertiK, 09/26/2024]:

The team mitigated this issue by restricting the minimum bet amount to 1000 BET and changes were reflected in the commit [cccc083c5d19a632180275ecc889ed7f40d1cd09](https://github.com/betfinio/hilo-contract/commit/cccc083c5d19a632180275ecc889ed7f40d1cd09). Besides, it's also recommended the team to keep enough funds in the subscription account to pay the Chainlink fee, ensuring all the requests can be handled.

HLT-02 | ENHANCED SECURITY THROUGH INCREASED BLOCK CONFIRMATIONS IN CHAINLINK VRF REQUESTS ON POLYGON NETWORK

Category	Severity	Location	Status
Design Issue	● Medium	src/HighLow.sol (09/25-8177fa): 38	● Resolved

Description

The `requestConfirmations` constant in the `HighLow` contract is set at 3. This parameter specifies the minimum number of block confirmations that Chainlink's VRF (Verifiable Random Function) service should wait before delivering randomness. This setting is crucial due to the occurrence of chain reorganizations, a scenario where blocks and their transactions are rearranged, leading to potential changes in the block content. This issue is particularly relevant for applications deployed on Polygon, an Ethereum scaling solution that utilizes a Proof of Stake (PoS) consensus mechanism. On Polygon, multiple validators may propose blocks at the same block height simultaneously. Network delays can result in these blocks being received at different times by different nodes, creating temporary forks. Observations from **Forked Blocks** indicate that there are over five reorganizations daily, with some extending beyond 3 blocks in depth. Given that BetFin is active on Polygon, there's a potential risk that the outcome of a HighLow game could change. Specifically, if the transaction requesting randomness from the VRF is shifted to another block due to a reorg, the resulting randomness—and consequently the game's outcome—could be altered.

```
uint16 public constant requestConfirmations = 3;
```

Recommendation

It's recommended to set a larger `requestConfirmations` value. For example, the value could be set based on the average depth of reorganizations observed, plus a buffer to account for deeper than usual reorgs.

Alleviation

[Betfin Team, 09/27/2024]:

Issue acknowledged. Changes have been reflected in the commit hash: <https://github.com/betfinio/hilo-contract/commit/cea0df2e46eeb1a8da260efcf08c3a05ff88a39e>.

HLU-01 | COMPILER ERROR

Category	Severity	Location	Status
Coding Issue	● Medium	src/HighLow.sol (09/20-ad1835): 163	● Resolved

Description

In the recent commit [ad183550821acc7887fb22aad19d9c8b2791969e](#), the team removed the `AccessControl` inheritance from the `HighLow` contract. Consequently, this modification led to a compilation failure due to an "Undeclared identifier" error concerning the `_msgSender()` function.

```
28 contract Dice is VRFConsumerBaseV2Plus, GameInterface, ReentrancyGuard {
```

```
158     function placeBet(  
159         address _player,  
160         uint256 _amount,  
161         bytes calldata _data  
162     ) external override returns (address betAddress) {  
163     @> require(address(core) == _msgSender(), "D05");
```

This error arises because the `_msgSender()` function, typically available through the `AccessControl` or `Context` classes in OpenZeppelin's libraries, is no longer inherited, thus it's unrecognized in the current contract's scope.

Recommendation

It's recommended to change the `_msgSender()` to `msg.sender` in the `placeBet` function of `Dice` contract.

Alleviation

[Betfin Team, 09/20/2024]:

Issue acknowledged. The team resolved this issue in the commit hash [cccc083c5d19a632180275ecc889ed7f40d1cd09](#) by changing the `_msgSender()` to `msg.sender`.

HLB-01 | UNCHECKED ERC-20 `transfer()` / `transferFrom()` CALL

Category	Severity	Location	Status
Volatile Code	● Minor	src/HighLow.sol (09/06-3c20e9): 157, 213-216	● 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.

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

[Betfin Team, 09/20/2024]:

Issue acknowledged. The team resolved this issue in the commit hash [cccc083c5d19a632180275ecc889ed7f40d1cd09](#) by checking the return value after transferring ERC-20 tokens.

HLB-03 | DISCREPANCY IN RANDOM NUMBER RANGE AND BETTING THRESHOLD IN `fulfillRandomWords` FUNCTION

Category	Severity	Location	Status
Logical Issue	● Minor	src/HighLow.sol (09/06-3c20e9): 136, 195	● Resolved

Description

The issue in the `fulfillRandomWords` function of the `HighLow` contract stems from a discrepancy between the range of generated random numbers and the range of the betting threshold, particularly when a player opts to bet on a higher outcome (`side` is true).

The random number (`value`) produced by the function falls within the range of [1, 9999], as determined by the modulo operation `(random % 9999) + 1`.

```
if ((value > threshold) == side) {  
    amount = getPossibleWin(threshold, side, bet.getAmount());  
}
```

Players are allowed to set a `threshold` for their bets within the same range, [1, 9999]. For a bet on a higher outcome to win, the generated random number (`value`) must exceed the `threshold`.

However, if a player selects the maximum `threshold` of 9999 and bets on the number being higher, winning is impossible. This is due to the maximum possible random number also being 9999. Consequently, there are no numbers within the range [1, 9999] that surpass 9999. This results in an unfair game condition where bets placed on the highest possible threshold (9999) with the expectation of a higher result are invariably destined to lose.

Recommendation

It's recommended to change the random number generation to cover a slightly broader range, such as [1, 10000], ensuring that a bet on a `threshold` of 9999 with `side` set to true has a potential to win.

Alleviation

[Betfin Team, 09/20/2024]:

Issue acknowledged. The team partially resolved this issue in the commit hash [352f43311a1895ffc2a433545206e694f8c1b86f](#) by changing the random number range to [1, 10000].

[CertiK, 09/23/2024]:

In the current design, where the range of random numbers is [1, 10000], players can employ a strategy to ensure a winning probability of 99.99%. For instance, a player could set the threshold to 1 for a 'High' bet, which guarantees a win if the random number falls between 2 and 10000. Similarly, setting the threshold to 9999 for a 'Low' bet ensures a win if the

random number is between 1 and 9999. When the probability of winning is this high, at 99.99%, the player receives a payout of $\text{BetAmount} * 10000/9999$. Although the profit per bet is small, it guarantees that the player does not lose money and can repeatedly place bets to accumulate profits. We would like to confirm if this design aligns with the intended outcome.

[Betfin Team, 09/25/2024]:

Issue acknowledged. The team resolved this issue in the commit hash [8177facb1cf5cc7f9e5472c87e83939a38e72bc](#) by changing the threshold range to [100, 9900].

[CertiK, 09/26/2024]:

The team updated the code to slightly decrease the edge probability and changes were reflected in the commit [8177facb1cf5cc7f9e5472c87e83939a38e72bc](#).

HLB-02 SECURITY RISK DUE TO PRESENCE OF TEST FUNCTION

FulfillRandomWords()

Category	Severity	Location	Status
Logical Issue	● Informational	src/HighLow.sol (09/06-3c20e9): 88-91	● Resolved

Description

The function `FulfillRandomWords()` found in the `Dice` contract is marked for testing purposes, intended to simulate the behavior of the `fulfillRandomWords` callback from the Chainlink VRF (Verifiable Random Function) service. This function allows manual input of random numbers for testing how the contract reacts to different random outcomes.

Keeping such a function in the production version of the contract, especially without appropriate access controls, poses a significant security risk. Malicious actors could exploit this function to manipulate game outcomes by providing selected random values, leading to potential loss of funds or unfair advantages.

```
/* For Test Purpose - will be removed when deployment */
// Public function to test fulfillRandomWords
function FulfillRandomWords(
    uint256 requestId,
    uint256[] calldata randomWords
) public {
    fulfillRandomWords(requestId, randomWords);
}
```

Recommendation

It is recommended to remove the `FulfillRandomWords()` function from the production environment.

Alleviation

[Betfin Team, 09/20/2024]:

Issue Acknowledged. The team resolved this issue in the commit hash [0cb63c1dd392cc8addce029ee44dd03b8b18a655](#) by removing the `FulfillRandomWords()` function.

OPTIMIZATIONS | BETFIN HIGH LOW CONTRACTS

ID	Title	Category	Severity	Status
HIG-01	Variables That Could Be Declared As Immutable	Gas Optimization	Optimization	● Resolved
HLT-01	State Variable Should Be Declared Constant	Gas Optimization	Optimization	● Resolved

HIG-01 | VARIABLES THAT COULD BE DECLARED AS IMMUTABLE

Category	Severity	Location	Status
Gas Optimization	● Optimization	src/HighLowBet.sol (09/06-3c20e9): 8, 9, 10	● 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

[Betfin Team, 09/20/2024]:

Issue Acknowledged. The team resolved this issue in the commit hash [ad183550821acc7887fb22aad19d9c8b2791969e](#) by declaring the variables as `immutable`.

HLT-01 | STATE VARIABLE SHOULD BE DECLARED CONSTANT

Category	Severity	Location	Status
Gas Optimization	● Optimization	src/HighLow.sol (09/25-8177fa): 41, 42	● Resolved

Description

State variables that never change should be declared as `constant` to save gas.

```
42     uint256 public MAX_THRESHOLD = 9900;
```

- `MAX_THRESHOLD` should be declared `constant`.

```
41     uint256 public MIN_THRESHOLD = 100;
```

- `MIN_THRESHOLD` should be declared `constant`.

Recommendation

We recommend adding the `constant` attribute to state variables that never change.

Alleviation

[Betfin Team, 09/27/2024]:

Issue acknowledged. Changes have been reflected in the commit hash: <https://github.com/betfinio/hilo-contract/commit/b483166f37cd0e5121bdad0bd17abe15e3005bae>.

APPENDIX | BETFIN HIGH LOW CONTRACTS

Finding Categories

Categories	Description
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.
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.
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.

