CERTIK

Code Security Assessment

Biswap (Audit 4)

Sept 10th, 2021

Table of Contents

Summary

This report has been prepared for Biswap (Audit 4) to discover issues and vulnerabilities in the source code of the Biswap (Audit 4) project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

Additionally, this audit is based on a premise that all external contracts were implemented safely.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Biswap (Audit 4)
Platform	Ethereum
Language	Solidity
Codebase	https://github.com/biswap-org/lottery/tree/master/contracts
Commit	3792597b5b4417baea6be71ea5a120740e40f62b 636156de65630591fcb08b333b254e48f5365947

Audit Summary

Delivery Date	Sept 10, 2021
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Mitigated	Resolved
● Critical	0	0	0	0	0	0	0
● Major	3	0	0	1	0	0	2
● Medium	0	0	0	0	0	0	0
● Minor	1	0	0	0	0	0	1
● Informational	7	0	0	0	0	0	7
● Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
LBC	Lottery.sol	84e36ba429c4e5654c2d1d35a5881471adde3527340f80bb9d14a1e276e0f0d4
RNG	RandomNumberGenerator.sol	3cd026582fe88882caeca4cd78a5a9ccb9741cccd71942cf50def2279a0b5519

Understandings

Overview

The `BiswapLottery` is a lottery contract. The winning numbers for lottery activities are randomly generated using chainlink.

The `operator` can start the lottery activity, including setting the activity period, the price of each lottery ticket, the discount of the activity. The activity is divided into 6 levels, the higher the level, the more bonus.

After the activity starts, users can use BSW tokens to purchase lottery tickets, and the single purchase limit is 100 tickets.

When the purchase period ends, the `operator` will draw the winning number and calculate the bonus. The total sales of each activity is divided into the following four parts:

- 13% of sales will be destroyed
- 7% of sales will send to referrals and competition
- bonus
- the remaining sales are determined by the `operator` to accumulate the bonus to the next activity or withdraw to the `treasury` account.

If the user wins, the user can claim the bonus during the settlement period.
The reward calculation process is not in the contract.
It should be noted that the `owner` has the authority to modify all the parameters mentioned above.

Privileged Functions

The contract contains the following privileged functions that are restricted by some modifiers. They are used to modify the contract configurations and address attributes. We grouped these functions below:

The `onlyOwner` modifier:

Contract `Ownable`:

- renounceOwnership()
- transferOwnership(address newOwner)

Contract `BiswapLottery`:

- changeRandomGenerator(address _randomGeneratorAddress)

- changeOracle(address _priceOracleAddress)
- recoverWrongTokens(address _tokenAddress, uint256 _tokenAmount)
- setMinAndMaxTicketPriceInBSW(uint256 _minPriceTicketInBSW, uint256 _maxPriceTicketInBSW)
- setMaxNumberTicketsPerBuy(uint256 _maxNumberTicketsPerBuy)
- setBurningAndCompetitionShare(uint256 _burningShare, uint256 _competitionAndRefShare)
- setMaxDiffPriceUpdate(uint256 _maxDiffPriceUpdate)
- setManagingAddresses(address _operatorAddress, address _treasuryAddress, address _injectorAddress, address _burningAddress, address _competitionAndRefAddress)

Contract `RandomNumberGenerator`:

- setFee(uint256 _fee)
- setKeyHash(bytes32 _keyHash)
- setLotteryAddress(address _biswapLottery)
- withdrawTokens(address _tokenAddress, uint256 _tokenAmount)

The `nonReentrant` modifier:

Contract `BiswapLottery`:

- buyTickets(uint256 _lotteryId, uint32[] calldata _ticketNumbers)
- claimTickets(uint256 _lotteryId, uint256[] calldata _ticketIds, uint32[] calldata _brackets)
- closeLottery(uint256 _lotteryId)
- drawFinalNumberAndMakeLotteryClaimable(uint256 _lotteryId, uint[6] calldata _bswPerBracket, uint[6] calldata _countTicketsPerBracket, bool _autoInjection)

The `onlyOperator` modifier:

Contract `BiswapLottery`:

- closeLottery(uint256 _lotteryId)
- drawFinalNumberAndMakeLotteryClaimable(uint256 _lotteryId, uint[6] calldata _bswPerBracket, uint[6] calldata _countTicketsPerBracket, bool _autoInjection)
- startLottery(uint256 _endTime, uint256 _priceTicketInUSDT, uint256 _discountDivisor, uint256[6] calldata _rewardsBreakdown)

The `notContract` modifier:

Contract `BiswapLottery`:

- buyTickets(uint256 _lotteryId, uint32[] calldata _ticketNumbers)

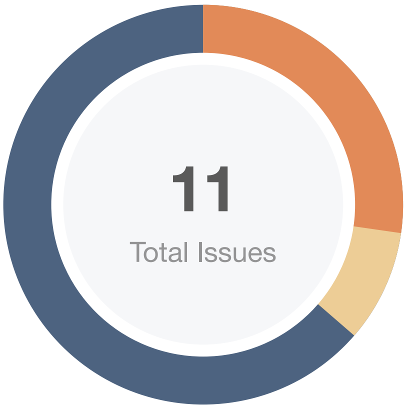
- claimTickets(uint256 _lotteryId, uint256[] calldata _ticketIds, uint32[] calldata _brackets)

The `onlyOwnerOrInjector` modifier:

Contract `BiswapLottery`:

- injectFunds(uint256 _lotteryId, uint256 _amount)

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**3** (27.27%)
🟨 **Medium**	**0** (0.00%)
🟨 **Minor**	**1** (9.09%)
🟦 **Informational**	**7** (63.64%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
LBC-01	Unlocked Compiler Version Declaration	Language Specific	● Informational	✓ Resolved
LBC-02	Check Effect Interaction Pattern Violated	Logical Issue, Coding Style	● Informational	✓ Resolved
LBC-03	Missing Input Validation	Logical Issue	● Informational	✓ Resolved
LBC-04	Missing Input Validation	Logical Issue	● Informational	✓ Resolved
LBC-05	Conflicting Requirements	Logical Issue	● Major	✓ Resolved
LBC-06	Meaningless State Variables	Coding Style, Logical Issue	● Informational	✓ Resolved
LBC-07	Missing Input Validation	Logical Issue	● Minor	✓ Resolved
LBC-08	Wrong Judgment Condition	Logical Issue	● Major	✓ Resolved
LBC-09	Variable Naming Error	Logical Issue, Coding Style	● Informational	✓ Resolved
LBC-10	Centralization Risk	**Centralization / Privilege**	● **Major**	ⓘ Acknowledged
RNG-01	Unlocked Compiler Version Declaration	Language Specific	● Informational	✓ Resolved

LBC-01 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	● Informational	Lottery.sol: 10, 35, 103, 166, 248, 465, 563, 584, 599, 678	✓ Resolved

Description

The compiler version utilized throughout the project uses the `^` prefix specifier, denoting that a compiler version that is greater than the version will be used to compile the contracts. It is recommended the compiler version be consistent throughout the codebase.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and thus be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-02 | Check Effect Interaction Pattern Violated

Category	Severity	Location	Status
Logical Issue, Coding Style	● Informational	Lottery.sol: 1065	✓ Resolved

Description

The order of external call/transfer and storage manipulation must follow check effect interaction pattern.

Recommendation

We advise client to check if storage manipulation is before the external call/transfer operation by considering following modification:

```
1065 function injectFunds(uint256 _lotteryId, uint256 _amount)
1066                   external override onlyOwnerOrInjector {
1067  require(_lotteries[_lotteryId].status == Status.Open, "Lottery not open");
1068 
1069  _lotteries[_lotteryId].amountCollectedInBSW += _amount;
1070  bswToken.safeTransferFrom(address(msg.sender), address(this), _amount);
1071 
1072  emit LotteryInjection(_lotteryId, _amount);
1073 }
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-03 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	● Informational	Lottery.sol: 812~813	✓ Resolved

Description

The given input is missing the sanity check for non-zero address in the aforementioned line.

Recommendation

We recommend adding the check for the passed-in values to prevent unexpected error as below: constructor():

```
812  require(_bswTokenAddress != address(0), "_bswTokenAddress address cannot be 0");
813  require(_usdtTokenAddress != address(0), "_usdtTokenAddress address cannot be 0");
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-04 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	● Informational	Lottery.sol: 1179	✓ Resolved

Description

`maxNumberTicketsPerBuyOrClaim` should be less than or equal to `MIN_DISCOUNT_DIVISOR`. If the private key of the `owner` account is leaked or the owner misuses and sets `maxNumberTicketsPerBuyOrClaim` to 301, then users will purchase the maximum number of lottery tickets for free.

Recommendation

We recommend adding the check for the passed-in value of `_maxNumberTicketsPerBuy` to prevent unexpected error.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-05 | Conflicting Requirements

Category	Severity	Location	Status
Logical Issue	● Major	Lottery.sol: 998, 1005	✓ Resolved

Description

According to our understanding, `rewardsBreakdown` represents the proportion of the bonus of each level to the total bonus, `amountToDistribute` represents the rewards that can be distributed (including the bonuses accumulated by the previous lottery), if the number of votes in each bracket is greater than 0, then the minimum value of `bswSumPerBrackets` is equal to `amountToDistribute`, causing these two requirements to become contradictory:

```
 998  require(
 999      winningPoolPerBracket >= (_lotteries[_lotteryId].rewardsBreakdown[i] * amountToDistribute) / 10000,
1000      'Wrong amount on bracket'
1001  );
```

```
1005  require(bswSumPerBrackets <= amountToDistribute, 'Wrong brackets Total amount');
```

Recommendation

We recommend modifying as below:

```
...
if(_countTicketsPerBracket[i] > 0){
    require(
        winningPoolPerBracket <=
            (_lotteries[_lotteryId].rewardsBreakdown[i] * amountToDistribute) / 10000,
        'Wrong amount on bracket');
}
...
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-06 | Meaningless State Variables

Category	Severity	Location	Status
Coding Style, Logical Issue	● Informational	Lottery.sol: 752	✓ Resolved

Description

Because `transformedWinningNumber` and `transformedUserNumber` are calculated in the same way, adding the same value does not affect the final comparison result, so `_bracketCalculator` state variable is meaningless.

Recommendation

We recommend removing `_bracketCalculator` state variable.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-07 | Missing Input Validation

Category	Severity	Location	Status
Logical Issue	● Minor	Lottery.sol: 1247	✓ Resolved

Description

The pass-in value of `_numberTickets` lacks verification. According to the calculation logic of the reward in the contract, if `_numberTickets = _discountDivisor + 1`, then the calculation result is 0.

Recommendation

We recommend adding the verification for `_numberTickets` to prevent unexpected error.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-08 | Wrong Judgment Condition

Category	Severity	Location	Status
Logical Issue	● Major	Lottery.sol: 1317	✓ Resolved

Description

The judgment condition for verifying the validity of the pass-in value of `_ticketId` should use the condition `||` instead of `&&`.

Recommendation

We recommend using `||` instead of `&&` as below:

```
1317  ...
1318  if ( _lotteries[_lotteryId].firstTicketIdNextLottery < _ticketId) ||
1319   _lotteries[_lotteryId].firstTicketId >= _ticketId ){
1320          return 0;
1321  }
1322  ...
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-09 | Variable Naming Error

Category	Severity	Location	Status
Logical Issue, Coding Style	● Informational	Lottery.sol: 1427, 1430	✓ Resolved

Description

`userNumber` represents the number selected by the user when purchasing the lottery ticket, `winningTicketNumber` represents the winning number of the lottery ticket, so the variable assignment is reversed.

Recommendation

We recommend modifying as below:

```
1427  uint32 winningTicketNumber = _lotteries[_lotteryId].finalNumber;
1428  uint32 userNumber = _tickets[_ticketId].number;
```

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

LBC-10 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	Lottery.sol	ⓘ Acknowledged

Description

In the contract `BiswapLottery`, the role `owner` has the authority over the following function:

1. modify `randomGenerator` address through `changeRandomGenerator` function.
2. modify `priceOracle` address through `changeOracle` function.
3. modify the minimum/maximum price of each tickets through `setMinAndMaxTicketPriceInBSW` function.
4. modify the maximum number of tickets for a single purchases through `setMaxNumberTicketsPerBuy` function.
5. modify `burningShare` and `competitionAndRefShare` through `setBurningAndCompetitionShare` function.
6. modify the maximum value of BSWToken price fluctuation through `maxDiffPriceUpdate` function.
7. modify `operatorAddress` ,`treasuryAddress` ,`injectorAddress` ,`burningAddress`,`competitionAndRefAddress` through `setManagingAddresses` function.

In the contract `BiswapLottery`, the role `operator` has the authority over the following function:

1. start a new lottery through `startLottery` function.
2. close the lottery through `closeLottery` function.
3. calculate lottery prizes through `drawFinalNumberAndMakeLotteryClaimable` function.

without obtaining the consensus of the community.

Recommendation

We advise the client to carefully manage the `owner`,`injector`,`operator` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at the different levels in terms of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;

- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

Customer response:
Immediately after the contract deployment, we will implement a 48-hour timelock on the owner role for awareness on privileged operations.

RNG-01 | Unlocked Compiler Version Declaration

Category	Severity	Location	Status
Language Specific	● Informational	RandomNumberGenerator.sol: 9, 34, 102, 184, 401, 499, 537, 579, 768, 789, 861	✓ Resolved

Description

The compiler version utilized throughout the project uses the `^` prefix specifier, denoting that a compiler version that is greater than the version will be used to compile the contracts. It is recommended the compiler version be consistent throughout the codebase.

Recommendation

It is a general practice to instead lock the compiler at a specific version rather than allow a range of compiler versions to be utilized to avoid compiler-specific bugs and thus be able to identify ones more easily. We recommend locking the compiler at the lowest possible version that supports all the capabilities wished by the codebase. This will ensure that the project utilizes a compiler version that has been in use for the longest time and as such is less likely to contain yet-undiscovered bugs.

Alleviation

The team heeded our advice and changed related codes. Code change was applied in commit 636156de65630591fcb08b333b254e48f5365947.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

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.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND “AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.