AUDITEK SECURITY

COMPLETE AUDIT REPORT

Security Assessment

AUTOCAKEAVAX

Sep 2st, 2021

Prepared by
ELLEN DOWNING

Approved by
LILLIAN PRATT

Table of Contents

Summary

This report has been prepared for AutoCakeAvax to discover issues and vulnerabilities in the source code of the AutoCakeAvax project as well as any contract dependencies that were not part of an ofcially 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 specifcations 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 fndings that ranged from critical to informational. We recommend addressing these fndings 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 verifed in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	AutoCakeAvax
Platform	BSC
Language	Solidity
Codebase	• 0xB1e55092fEbb6b830c265D6B0E33514B963a7A68 • 0xB1e55092fEbb6b830c265D6B0E33514B963a7A68 • https://bscscan.com/address/ 0xB1e55092fEbb6b830c265D6B0E33514B963a7A68
Commit	

Audit Summary

Delivery Date	Sep 2st, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	✓ Resolved
● Critical	0	0	0	0	0	0
● Major	2	0	0	0	0	2
● Medium	3	0	0	1	0	2
● Minor	3	0	0	3	0	0
● Informational	8	0	0	3	0	5
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
ACA	AutoCakeAvax.sol	01fb4807fba0eda416a41fc03f78f4bb44dd1e78490193357fd29dcadada7ead

Review Notes

Dependencies

There are a few depending injection contracts or addresses in the current project:

- `marketingAddress` , `charityAddress` , `buybackAddress` , and PancakeSwap Router address for the contract `AutoCakeAvax` .

We assume these contracts or addresses are valid and non-vulnerable actors and implementing proper logic to collaborate with the current project.

Privileged Functions

The `owner` role of the contract `AutoCakeAvax` can operate on the many privileged functions as we grouped below.

Account management functions for inclusion and exclusion in the fee and reward system:

- `AutoCakeAvax.excludeFromReward(address)`
- `AutoCakeAvax.includeInReward(address)`
- `AutoCakeAvax.excludeFromFee(address)`
- `AutoCakeAvax.includeInFee(address)`

Modifcation of fees, threshold token amount, and proportion of BNB allocation:

- `AutoCakeAvax.setTaxFeePercent(uint256, uint256)`
- `AutoCakeAvax.setMaxTxAmount(uint256)`
- `AutoCakeAvax.setCharityDivisor(uint256)`
- `AutoCakeAvax.setMarketingDivisor(uint256)`
- `AutoCakeAvax.setBuyBackDivisor(uint256)`
- `AutoCakeAvax.setLPDivisor(uint256)`
- `AutoCakeAvax.setNumTokensSellToAddToLiquidity(uint256)`

Confguration of charity, buyback, and marketing addresses:

- `AutoCakeAvax.setCharityAddress(address)`
- `AutoCakeAvax.setBuyBackAddress(address)`
- `AutoCakeAvax.setMarketingAddress(address)`

Toggle feature of the LP acquisition mechanism and sale phases:

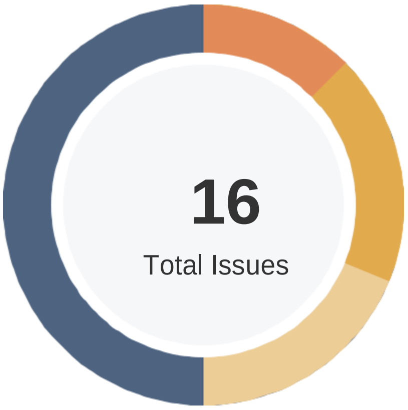
- [redacted]
- [redacted]
- [redacted]
- [redacted]

Manipulation of the contract ownership:

- [redacted]
- [redacted]

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notifed to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the [redacted] contract.

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**2** (12.50%)
🟨	**Medium**	**3** (18.75%)
🟨	**Minor**	**3** (18.75%)
🟦	**Informational**	**8** (50.00%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
ACA-01	Centralization Risk in `addLiquidity`	**Centralization / Privilege**	● **Major**	✓ Resolved
ACA-02	Disproportionate BNB Allocation and Liquidity Tokens	Logical Issue	● Major	✓ Resolved
ACA-03	Centralization Risk	**Centralization / Privilege**	● **Medium**	ⓘ Acknowledged
ACA-04	Non-Guaranteed Sum of Divisors	Logical Issue	● Medium	✓ Resolved
ACA-05	Possible to Gain Ownership after Renouncing the Contract Ownership	**Logical Issue, Centralization / Privilege**	● **Medium**	✓ Resolved
ACA-06	Incorrect Error Message	Logical Issue	● Minor	ⓘ Acknowledged
ACA-07	Third Party Dependencies	Control Flow	● Minor	ⓘ Acknowledged
ACA-08	Potential Sandwich Attack	Volatile Code	● Minor	ⓘ Acknowledged
ACA-09	Typo in the Contract	Coding Style	● Informational	✓ Resolved
ACA-10	Return Value Not Handled	Volatile Code	● Informational	ⓘ Acknowledged
ACA-11	Missing Event Emitting	Coding Style	● Informational	ⓘ Acknowledged

ID	Title	Category	Severity	Status
ACA-12	Unused Event	Coding Style	● Informational	ⓘ Acknowledged
ACA-13	Variable Could Be Declared As `constant`	Gas Optimization	● Informational	✓ Resolved
ACA-14	Function and Variable Naming Doesn't Match the Operating Environment	Coding Style	● Informational	✓ Resolved
ACA-15	Division Before Multiplication	Language Specifc	● Informational	✓ Resolved
ACA-16	Liquidity Can Only Be Added When Selling the Token	Logical Issue	● Informational	✓ Resolved

ACA-01 | Centralization Risk i[redacted]

Category	Severity	Location	Status
Centralization / Privilege	● **Major**	AutoCakeAvax.sol: 711	✓ Resolved

Description

The [redacted] function calls the [redacted] function listed below with the [redacted] address specified as [redacted] for acquiring the generated LP tokens from the [redacted] pool. As a result, over time the [redacted] address will accumulate a significant portion of LP tokens. If the [redacted] is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

```
// add the liquidity
uniswapV2Router.addLiquidityETH{value: ethAmount}(
    address(this),
    tokenAmount,
    0, // slippage is unavoidable
    0, // slippage is unavoidable
    owner(),
    block.timestamp
);
```

Recommendation

We advise the [redacted] address of the [redacted] function call to be replaced by the contract itself, i.e. [redacted], and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the [redacted] account is compromised. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or via smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- 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

The client heeded our advice and resolved this issue by setting the recipient of the liquidity tokens to `address(this)`. The fxing is refected in the code deployed at `0xcb5f9b3f12aef688416d1405dff0e45b591fa6bc`.

ACA-02 | Disproportionate BNB Allocation and Liquidity Tokens

Category	Severity	Location	Status
Logical Issue	● Major	AutoCakeAvax.sol: 654~672	Resolved

Description

Assume all parameters are set to default values. In function [redacted] , $\frac{7}{8}$ [redacted] ACAZ tokens is used to swap for BNB and the rest $\frac{1}{8}$ ACAZ tokens are used to add liquidity to the 'ACAZ-BNB' pool. Specifcally, $\frac{3}{7}$ of the swapped BNB is evenly transferred to the charity, marketing, and buyback address; $\frac{1}{14}$ of the swapped BNB is used to add liquidity to the 'ACAZ-BNB' pool. Here raise the following two major concerns:

1. There will be half of the swapped BNB accumulated in the contract for each call of function [redacted] . Meanwhile, the contract does not appear to provide a way to withdraw those BNB, and they will be locked in the contract forever.
2. While adding liquidity, the value of 'tokenAmountForLP' ACAZ tokens (i.e., $\frac{1}{8}$ of [redacted] ACAZ tokens) does not match that of [redacted] BNB (i.e., $\frac{1}{14}$ of the swapped BNB).

We expect the ACAZ team to provide more details about their design in terms of allocating swapped BNB and adding liquidity.

Alleviation

The client updated the allocation of swapped BNB. Specifcally, $\frac{6}{7}$ of the swapped BNB is evenly transferred to the charity, marketing, and buyback address; $\frac{1}{7}$ of the swapped BNB is used to add liquidity to the 'ACAZ-BNB' pool. The fxing is refected in the code deployed at [redacted].

ACA-03 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	● Medium	AutoCakeAvax.sol: 585, 595, 866, 870, 874, 879, 883, 887, 891, 895, 901, 905, 908, 911, 915, 920, 924, 931	ⓘ Acknowledged

Description

With the modifer [redacted], the 'owner' has the authority to call the following 18 sensitive functions to change the settings of the [redacted] contract.

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

Any compromise to the [redacted] account may allow the hacker to adversarially manipulate the settings of the contract.

Recommendation

We advise the client to carefully manage the 'owner' 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 via smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Timelock 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

N/A

ACA-04 | Non-Guaranteed Sum of Divisors

Category	Severity	Location	Status
Logical Issue	● Medium	AutoCakeAvax.sol: 934	✓ Resolved

Description

[redacted] should equal the sum of [redacted], [redacted], [redacted] [redacted] . However, it is not guaranteed that the sum would be 12 because these divisors can be modifed [redacted] [redacted] , [redacted] and [redacted] .

Recommendation

We recommend setting [redacted] to be the sum of [redacted], [redacted] [redacted] and [redacted] instead of the constant 12.

Alleviation

The client heeded our advice and resolved this issue by setting [redacted] to be the sum of [redacted], [redacted], [redacted] and [redacted] . The fxing is refected in the code deployed at [redacted].

ACA-05 | Possible to Gain Ownership after Renouncing the Contract Ownership

Category	Severity	Location	Status
Logical Issue, Centralization / Privilege	● **Medium**	AutoCakeAvax.sol: 189	✓ Resolved

Description

An owner is possible to gain ownership of the contract even if he/she calls the function [redacted] to renounce the ownership. This can be achieved by performing the following operations:

1. Call [redacted] to lock the contract. The variable [redacted] is set to the current owner.
2. Call [redacted] to unlock the contract.
3. Call [redacted] to leave the contract without an owner.
4. Call [redacted] to regain ownership.

Recommendation

We advise updating/removing [redacted] and [redacted] functions in the contract, or removing the [redacted] if such a privilege retains at the protocol level. If timelock functionality could be introduced, we recommend using the implementation of Compound fnance as a reference.

Reference: https://github.com/compound-fnance/compound-protocol/blob/master/contracts/Timelock.sol

Alleviation

The client heeded our advice and resolved this issue by removing the function [redacted] and [redacted] . Meanwhile, they also removed [redacted] and [redacted]. The fxing is refected in the code deployed at [redacted].

ACA-06 | Incorrect Error Message

Category	Severity	Location	Status
Logical Issue	● Minor	AutoCakeAvax.sol: 596	ⓘ Acknowledged

Description

The error message in `require(_isExcluded[account], "Account is already excluded")` does not describe the error correctly.

Recommendation

The message "Account is already excluded" should be changed to "Account is not excluded" .

Alleviation

N/A

ACA-07 | Third Party Dependencies

Category	Severity	Location	Status
Control Flow	● Minor	AutoCakeAvax.sol: 484	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party PancakeSwap protocols. The scope of the audit treats third-party entities as black boxes and assumes their functional correctness. However, in the real world, third-party entities can be compromised and this may lead to lost or stolen assets. In addition, upgrades of third-party entities can possibly create severe impacts, such as increasing fees of third-party entities, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of the ACAZ protocol requires the interaction PancakeSwap protocol for adding liquidity to the ACAZ-BNB pool and swapping tokens. We encourage the team to constantly monitor the statuses of those third-party entities to mitigate the side efects when unexpected activities are observed.

Alleviation

N/A

ACA-08 | Potential Sandwich Attack

Category	Severity	Location	Status
Volatile Code	● Minor	AutoCakeAvax.sol: 680~686, 701~706, 716~723	ⓘ Acknowledged

Description

Potential sandwich attacks could happen if calling `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens`, `uniswapV2Router.swapExactTokensForETHSupportingFeeOnTransferTokens`, and `uniswapV2Router.addLiquidityETH` without setting restrictions on slippage.

For example, when we want to make a transaction of swapping 100 AToken for 1 ETH, an attacker could raise the price of ETH by adding AToken into the pool before the transaction so we might only get 0.1 ETH. After the transaction, the attacker would be able to withdraw more than he deposited because the total value of the pool increases by 0.9 ETH.

Recommendation

We recommend setting reasonable minimum output amounts, instead of 0, based on token prices when calling the aforementioned functions.

Alleviation

N/A

ACA-09 | Typo in the Contract

Category	Severity	Location	Status
Coding Style	● Informational	AutoCakeAvax.sol: 471	✓ Resolved

Description

The variable name `tokensIntoLiqudity` should be `tokensIntoLiquidity` .

Recommendation

We recommend correcting this typo in the contract.

Alleviation

The client heeded our advice and resolved this issue by changing `tokensIntoLiqudity` to `tokensIntoLiquidity`. The fxing is refected in the code deployed at `0xcb5f9b3f12aef688416d1405dff0e45b591fa6bc`.

ACA-10 | Return Value Not Handled

Category	Severity	Location	Status
Volatile Code	● Informational	AutoCakeAvax.sol: 716	ⓘ Acknowledged

Description

The return values of function [redacted] are not properly handled.

```
        uniswapV2Router.addLiquidityETH{value: ethAmount}(
            address(this),
            tokenAmount,
            0, // slippage is unavoidable
            0, // slippage is unavoidable
            owner(),
            block.timestamp
        );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

N/A

ACA-11 | Missing Event Emitting

Category	Severity	Location	Status
Coding Style	● Informational	AutoCakeAvax.sol: 585, 595, 866, 870, 874, 879, 883, 887, 891, 895, 901, 905, 908, 911, 920, 924, 931	ⓘ Acknowledged

Description

In the contract [redacted]vax , there are a bunch of functions that can change state variables. However, these functions (as listed below) do not emit events to pass the changes out of the chain:

- [redacted]ss)
- [redacted]ss)
- [redacted]ss)
- [redacted]ss)
- [redacted]56)
- [redacted]56)
- [redacted]56)
- [redacted]56)
- [redacted]56)
- [redacted]56)
- [redacted]56)
- [redacted]ss)
- [redacted]ss)
- [redacted]ss)
- [redacted]ol)
- [redacted]()
- [redacted]()

Recommendation

We recommend declaring and emitting corresponding events for all the essential state variables that are possible to be changed during runtime.

Alleviation

N/A

ACA-12 | Unused Event

Category	Severity	Location	Status
Coding Style	● Informational	AutoCakeAvax.sol: 468, 469, 471	ⓘ Acknowledged

Description

The events of `RewardLiquidityProviders` , `BuyBackEnabledUpdated` , and `SwapAndLiquify` are declared but never used.

Recommendation

We recommend removing these events or emitting them in the right places.

Alleviation

N/A

ACA-13 | Variable Could Be Declared A[redacted]

Category	Severity	Location	Status
Gas Optimization	● Informational	AutoCakeAvax.sol: 435, 439~441	✓ Resolved

Description

Variables [redacted] , [redacted] , [redacted] , an[redacted] could be declared a[redacted] since these state variables are never to be changed.

Recommendation

We recommend declaring those variables as [redacted] .

Alleviation

The client heeded our advice and resolved this issue by declaring aforementioned variables as [redacted]. The fxing is refected in the code deployed at [redacted]

ACA-14 | Function and Variable Naming Doesn't Match the Operating Environment

Category	Severity	Location	Status
Coding Style	● Informational	AutoCakeAvax.sol: 460, 674, 694	✓ Resolved

Description

The AutoCakeAvax contract uses PancakeSwap for swapping and adding liquidity to the PancakeSwap pool, but naming it Uniswap. Fun[redacted]int256) swaps ACAZ token for BNB instead of ETH, and similily fun[redacted]int256) swaps BNB instead of ETH for ACAZ token.

Recommendation

We recommend changing "Uniswap" and "ETH" to "PancakeSwap" and "BNB" in the contract respectively to match the operating environment and avoid confusion.

Alleviation

The client heeded our advice and resolved this issue by changing "Uniswap" and "ETH" to "PancakeSwap" and "BNB", respectively, in the contract. The fxing is refected in the code deployed at [redacted].

ACA-15 | Division Before Multiplication

Category	Severity	Location	Status
Language Specifc	● Informational	AutoCakeAvax.sol: 655, 661, 663, 665, 667	✓ Resolved

Description

In function [redacted] , the division operations are performed before the multiplication operations, while performing multiplication before division can sometimes reduce or avoid loss of precision.

Recommendation

We recommend performing multiplication before division to avoid the loss of precision.

Alleviation

The client heeded our advice and resolved this issue by performing multiplication before division in aforementioned places. The fxing is refected in the code deployed at [redacted].

ACA-16 | Liquidity Can Only Be Added When Selling the Token

Category	Severity	Location	Status
Logical Issue	● Informational	AutoCakeAvax.sol: 637	✓ Resolved

Description

To add liquidity, the condition `to == uniswapV2Pair` needs to be true. That is, assuming the rest three conditions are true, liquidity can only be added if and only if the current transaction sells the ACAZ token. We expect the ACAZ team to confrm whether this is the intended design.

Alleviation

The client removed the condition `to == uniswapV2Pair`. Therefore, adding liquidity does not depend on the sell transaction anymore. The fxing is refected in the code deployed at `0xcb5f9b3f12aef688416d1405dff0e45b591fa6bc`.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege fndings 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 fndings do not afect the functionality of the code but generate diferent, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Logical Issue

Logical Issue fndings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow fndings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code fndings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Language Specific

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

Coding Style

Coding Style fndings usually do not afect 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" feld 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 fle hosted in the listed source repository under the specifed commit.

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

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 AuditeK'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 AuditeK 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. AuditeK's position is that each company and individual are responsible for their own due diligence and continuous security. AuditeK'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 AuditeK 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, AUDITEK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, AUDITEK 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, AUDITEK 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, AUDITEK 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 AUDITEK NOR ANY OF AUDITEK’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. AUDITEK 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 AUDITEK’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 AUDITEK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF AUDITEK 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 AUDITEK 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 2021 by leading academics in the feld of Computer Science from both Yale and Columbia University, AuditeK 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.