Security Assessment

Sugarland - Audit

Apr 2nd, 2022

Table of Contents

Summary

This report has been prepared for Sugarland - Audit to discover issues and vulnerabilities in the source code of the Sugarland - Audit 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 smart contracts are safely implemented.

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	Sugarland - Audit
Platform	BSC
Language	Solidity
Codebase	https://github.com/Sugarland-Coin/contracts https://bscscan.com/address/0x59eB96F0B6f5838021f0E8f412Afe65D1Bf44A02#code
Commit	13e2aeb9b84336c2f8ee45a8385718f25eff68a5

Audit Summary

Delivery Date	Apr 02, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

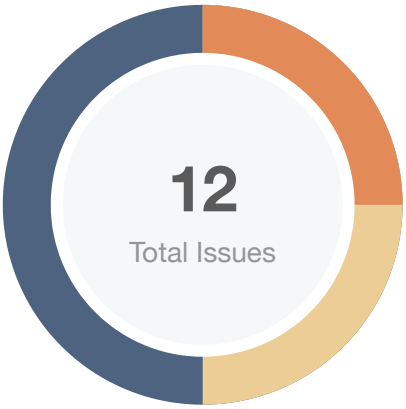
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	3	0	0	3	0	0	0
Medium	0	0	0	0	0	0	0
Minor	3	0	0	3	0	0	0
Informational	6	0	0	2	0	0	4
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
SSC	Sugarland.sol	98814f898580c479d2f4c9995d4918d25082eea3fc6efe358381a233ed2a87ef

Findings



Critical	0 (0.00%)
Major	3 (25.00%)
Medium	0 (0.00%)
Minor	3 (25.00%)
Informational	6 (50.00%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
GLOBAL-02	Third Party Dependencies	Volatile Code	Minor	Acknowledged
SSC-01	Initial Token Distribution	**Centralization / Privilege**	**Major**	Acknowledged
SSC-02	Centralized Risk In `addLiquidity`	**Centralization / Privilege**	**Major**	Acknowledged
SSC-03	Potential Sandwich Attacks	Logical Issue	Minor	Acknowledged
SSC-04	Unchecked Value of ERC-20 `transfer()`/`transferFrom()` Call	Volatile Code	Minor	Acknowledged
SSC-05	Mutability Specifiers Missing	Gas Optimization	Informational	Acknowledged
SSC-06	Unlocked Compiler Version	Language Specific	Informational	Resolved
SSC-07	Incorrect Argument Value	Logical Issue	Informational	Resolved
SSC-08	Fee Range	Logical Issue	Informational	Resolved
SSC-09	Missing Input Validation	Volatile Code	Informational	Resolved
SSC-10	Ambiguous Ratio Value For `swapAndLiquify()`	Logical Issue	Informational	Acknowledged

GLOBAL-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Acknowledged

Description

In the contract `Sugarland`, the role `owner` has authority over the following functions:

- `renounceOwnership()`
- `transferOwnership()`
- `excludeFromReward()`
- `includeInReward()`
- `excludeFromFee()`
- `includeInFee()`
- `setTransferTaxes()`
- `setBuyTaxes()`
- `setSellTaxes()`
- `startTrading()`
- `updateMarketingWallet()`
- `updateMaxTxAmount()`
- `updateMaxWalletBalance()`
- `updateSwapTokensAtAmount()`
- `updateSwapEnabled()`
- `updateRouterAndPair()`
- `setIsBot()`
- `rescueBNB()`
- `rescueAnyBEP20Tokens()`

Additionally, `marketingWallet` will be used to receive marketing fee.

Any compromise to these accounts may allow a hacker to take advantage of this authority.

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., multi-signature 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.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

Alleviation

The client gave the following response:

1. `New Deployed Contract has the option to send LP Tokens to a separate LP Recipient that's secured as a 2/3 MultiSig Wallet. And the Multisig wallet information will be posted on the website post-launch.`
2. `Contract Cannot be renounced and we do not have TimeLock but we are assuring the Safety of Dev Wallet & Private Keys associated with that wallet with only 1 Doxxed Founder.`
3. `Sugarland project has DAO Feature, to be soon activated with the launch of our Citizens NFT Collection and will have Community Polls for taking major steps.`

GLOBAL-02 | Third Party Dependencies

Category	Severity	Location	Status
Volatile Code	Minor	Global	Acknowledged

Description

The contract is serving as the underlying entity to interact with third-party swap protocols. The scope of the audit treats 3rd party entities as black boxes and assumes their functional correctness. However, in the real world, 3rd parties can be compromised and this may lead to lost or stolen assets. In addition, upgrades of 3rd parties can possibly create severe impacts, such as increasing fees of 3rd parties, migrating to new LP pools, etc.

Recommendation

We understand that the business logic of the `Sugarland` contract requires interaction with the swap protocol. We encourage the team to constantly monitor the statuses of 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

The client acknowledged this issue.

SSC-01 | Initial Token Distribution

Category	Severity	Location	Status
Centralization / Privilege	**Major**	Sugarland.sol: 182	Acknowledged

Description

All of the tokens are sent to the contract deployer when deploying the contract. This could be a centralization risk as the deployer can distribute tokens without obtaining the consensus of the community.

Recommendation

We recommend the team be transparent regarding the initial token distribution process, and the team shall make enough efforts to restrict the access of the private key.

Alleviation

The client gave the following response:

`Token Supply will be generated & added to Deployer Wallet and airdropped further to Migrating Wallets as per Public data & rest will be locked in DxLockers with vesting schedules as mentioned on the Official Website.`

`To maintain security, Private Keys will remain with only the Lead Doxxed Founder of the Project.`

SSC-02 | Centralized Risk In `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	Major	Sugarland.sol: 487	Acknowledged

Description

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

The `addLiquidity` function calls the `router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `Sugar-BNB` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens. If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, 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 `_owner` 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, f.e. Multisignature wallets.

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

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent 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 gave the following response:

```
We have introduced LP recipient and it protected with MultiSignature & transparent with
Community.
```

SSC-03 | Potential Sandwich Attacks

Category	Severity	Location	Status
Logical Issue	Minor	Sugarland.sol: 482~489, 501~507	Acknowledged

Description

A sandwich attack might happen when an attacker observes a transaction swapping tokens or adding liquidity without setting restrictions on slippage or minimum output amount. The attacker can manipulate the exchange rate by frontrunning (before the transaction being attacked) a transaction to purchase one of the assets and make profits by back running (after the transaction being attacked) a transaction to sell the asset.

The following functions are called without setting restrictions on slippage or minimum output amount, so transactions triggering these functions are vulnerable to sandwich attacks, especially when the input amount is large:

- `addLiquidity()`
- `swapTokensForBNB()`

Recommendation

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

Alleviation

The client gave the following response:

```
We will need to redeploy the contract, which won't be possible as of now.
```

SSC-04 | Unchecked Value Of ERC-20 `transfer()` / `transferFrom()` Call

Category	Severity	Location	Status
Volatile Code	Minor	Sugarland.sol: 554	Acknowledged

Description

The linked `transfer()`/`transferFrom()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

Recommendation

As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

Alleviation

The client gave the following response:

`we can choose not to rectify that Require Code Line because Metamask will not let it transact if there is an error.`

SSC-05 | Mutability Specifiers Missing

Category	Severity	Location	Status
Gas Optimization	Informational	Sugarland.sol: 126	Acknowledged

Description

The linked variables are assigned only once, either during their contract-level declaration or during the `constructor`'s execution.

Recommendation

For the former, we advise that the `constant` keyword is introduced in the variable declaration to greatly optimize the gas cost involved in utilizing the variable. For the latter, we advise that the `immutable` mutability specifier is set at the variable's contract-level declaration to greatly optimize the gas cost of utilizing the variables. Please note that the `immutable` keyword only works in Solidity versions `v0.6.5` and up.

Alleviation

The client acknowledged this issue.

SSC-06 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	Sugarland.sol: 2	Resolved

Description

The contract has unlocked compiler version. An unlocked compiler version in the source code of the contract permits the user to compile it at or above a particular version. This, in turn, leads to differences in the generated bytecode between compilations due to differing compiler version numbers. This can lead to an ambiguity when debugging as compiler specific bugs may occur in the codebase that would be hard to identify over a span of multiple compiler versions rather than a specific one.

Recommendation

We advise that the compiler version is instead locked at the lowest version possible that the contract can be compiled at. For example, for version `v0.8.10` the contract should contain the following line:

```
pragma solidity 0.8.10;
```

Alleviation

The development team solved this issue at https://bscscan.com/address/0x59eB96F0B6f5838021f0E8f412Afe65D1Bf44A02#code.

SSC-07 | Incorrect Argument Value

Category	Severity	Location	Status
Logical Issue	Informational	Sugarland.sol: 253	Resolved

Description

The statement at L253 is used to get `valuesFromGetValues` instance by calling `_getValues()` without fee. So the second argument `takeFee` should be passed-in `false`.

Recommendation

We advise refactoring the linked statement as below:

```
253  valuesFromGetValues memory s = _getValues(tAmount, false, code);
```

Alleviation

The development team solved this issue at https://bscscan.com/address/0x59eB96F0B6f5838021f0E8f412Afe65D1Bf44A02#code.

SSC-08 | Fee Range

Category	Severity	Location	Status
Logical Issue	Informational	Sugarland.sol: 303~311	Resolved

Description

The transaction fee should be only a small proportion of the transaction amount. So the transaction fee should be limited to a reasonable range when the linked functions are called.

Recommendation

We advise adding `require()` statements to restrict the value of the arguments, `_rfi`, `_marketing`, and `_liquidity`.

Alleviation

The development team solved this issue at https://bscscan.com/address/0x59eB96F0B6f5838021f0E8f412Afe65D1Bf44A02#code. The upper limit for each of the three types of fees, transfer/buy/sell, is set at 50%.

SSC-09 | Missing Input Validation

Category	Severity	Location	Status
Volatile Code	Informational	Sugarland.sol: 516~518	Resolved

Description

The given input is missing the check for the non-zero address.

Recommendation

We advise adding the check for the passed-in values to prevent unexpected errors as below:

```
516  function updateMarketingWallet(address _marketingWallet) external onlyOwner{
517      require(address(0) != _marketingWallet, "set marketing wallet to the zero
address");
518      marketingWallet = _marketingWallet;
519  }
```

Alleviation

The development team solved this issue at

https://bscscan.com/address/0x59eB96F0B6f5838021f0E8f412Afe65D1Bf44A02#code.

SSC-10 | Ambiguous Ratio Value For `swapAndLiquify()`

Category	Severity	Location	Status
Logical Issue	Informational	Sugarland.sol: 417~419, 453	Acknowledged

Description

The `swapAndLiquify()` functions can be called in the case of `Sugar` token transfer and sale. The taxes on `Sugar` token transfer and sale are different. The `swapAndLiquify()`, however, directly use the sales tax fee.

Recommendation

Please provide more information about the design logic.

Alleviation

The client gave the following response:

```
We acknowledge it and we want it in the way we have done.
```

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.

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.

Language Specific

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

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.