CERTIK

Security Assessment

Subgenix

May 3rd, 2022

Table of Contents

Summary

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

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	Subgenix
Platform	Avalanche
Language	Solidity
Codebase	https://github.com/Subgenix-Research/Core
Commit	5391e18302bfb290893f22c76919be84b0fbbfdc

Audit Summary

Delivery Date	May 03, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

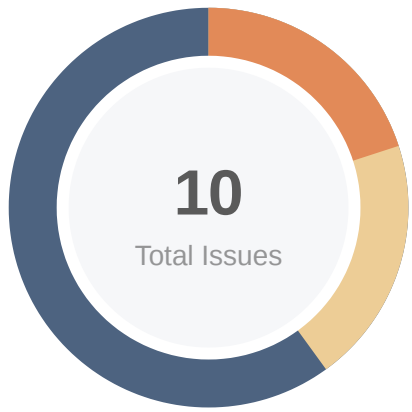
Vulnerability Summary

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

Audit Scope

ID	File	SHA256 Checksum
CKP	projects/contracts/interfaces	
ICK	projects/contracts/interfaces/Isgx.sol	22ae0cd78dd28b21fde85423b6243deeec1710cd072d1fffdd10c7531d7367ef
GSG	projects/contracts/GovernanceSGX.sol	263880dee09f91bf153e7ef4226b8903cbad5c81a50e583dfbd5a51397937e43
ISG	projects/contracts/interfaces/IgSGX.sol	d8f1c9640313114af9db84ab1f21cf4d6de7846a7ba012592aa3e3ca3d39bbac
IJC	projects/contracts/interfaces/IJoeRouter02.sol	8115d982f3841117e14f73f2e59faac0ca354cf50720ad1d794a0020c7ecf587
HCK	projects/contracts/lockupHell.sol	d55e2058fc420597c4f7fa3023b8b98a3f31ee03484d09aabb19151a39d23c4f
SCK	projects/contracts/Subgenix.sol	8f59c41a6b9fa05e4af5764983c52338ce8ca0a98da1fb9a0d7ba7661c062121
IJR	projects/contracts/interfaces/IJoeRouter01.sol	18f7aa98d1752530728a471c9c139e3a8d719de3ef8985b2959a99cc16dba29a
VFC	projects/contracts/VaultFactory.sol	1881d4a728f2ceb2234011f4418a36cd89f6b5c0c37bfa53caecfaaa611aca12
ILH	projects/contracts/interfaces/ILockupHell.sol	e13250ef6a8aba5998b572126aed2289eded57716d840526fa482a1350eb5278
CON	projects/contracts	

Findings



Critical	**0** (0.00%)
Major	**2** (20.00%)
Medium	**0** (0.00%)
Minor	**2** (20.00%)
Informational	**6** (60.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CON-01	Missing Zero Address Validation	Volatile Code	Minor	Resolved
CON-02	Unlocked Compiler Version	Language Specific	Informational	Resolved
GIT-01	Centralization Risk In VaultFactory.sol	**Centralization / Privilege**	**Major**	Acknowledged
SCK-01	Centralization Risk In Subgenix.sol	**Centralization / Privilege**	**Major**	Acknowledged
VFC-01	Missing `require` Statement	Volatile Code	Minor	Resolved
VFC-02	Misleading Comments	Coding Style	Informational	Resolved
VFC-03	Unclear Use Of `approve()`	Inconsistency	Informational	Resolved
VFC-04	Modifier Does Not Revert Transactions	Language Specific	Informational	Resolved
VFC-05	Unused Return Value	Volatile Code	Informational	Resolved
VFC-06	Function Should Be Declared External	Gas Optimization	Informational	Resolved

CON-01 | Missing Zero Address Validation

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/VaultFactory.sol (Base): 48, 49, 50, 51, 52, 53; projects/contracts/lockupHell.sol (Base): 120, 337	Resolved

Description

Addresses should be checked before assignment or external call to make sure they are not zero addresses.

File: projects/contracts/VaultFactory.sol (Line 48, Function `VaultFactory.constructor`)

```
        wavax = _wavax;
```

- `_wavax` is not zero-checked before being used.

File: projects/contracts/VaultFactory.sol (Line 49, Function `VaultFactory.constructor`)

```
        sgx = _sgx;
```

- `_sgx` is not zero-checked before being used.

File: projects/contracts/VaultFactory.sol (Line 50, Function `VaultFactory.constructor`)

```
        gSGX = _gSGX;
```

- `_gSGX` is not zero-checked before being used.

File: projects/contracts/VaultFactory.sol (Line 51, Function `VaultFactory.constructor`)

```
        treasury = _treasury;
```

- `_treasury` is not zero-checked before being used.

File: projects/contracts/VaultFactory.sol (Line 52, Function `VaultFactory.constructor`)

```
        research = _research;
```

- `_research` is not zero-checked before being used.

File: projects/contracts/VaultFactory.sol (Line 53, Function `VaultFactory.constructor`)

```
        lockup = _lockup;
```

- `_lockup` is not zero-checked before being used.

File: projects/contracts/lockupHell.sol (Line 120, Function `LockupHell.constructor`)

```
        sgx = sgxAddress;
```

- `sgxAddress` is not zero-checked before being used.

File: projects/contracts/lockupHell.sol (Line 337, Function `LockupHell.setVaultFactory`)

```
        vaultFactory = vaultAddress;
```

- `vaultAddress` is not zero-checked before being used.

Recommendation

The core business logic in `VaultFactory.sol` highly depends that the addresses were set correctly. We advise adding a zero-check for the passed-in address value to prevent unexpected errors.

Alleviation

[CertiK -4/11/22] The Subgenix team partially resolved the issue. The team included an `if-statement` that ensures the linked variables in `VaultFactory.sol` are not the zero address. The same resolution should be applied to the addresses in the `lockuphell.sol` contract. The commits can be seen on lines 54-60 at https://github.com/Subgenix-Research/Core/commit/b0942a1f15541c87f070e07bdad896aa7de3146c

[CertiK - 4/21/22] - The Subgenix team resolved the issue entirely adding the additional check in the `lockuphell.sol` contracts. The changes can be seen at this commit https://github.com/Subgenix-Research/Core/commit/6e73dffe2c1c3584eb935ec2c3531a775428a245

CON-02 | Unlocked Compiler Version

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/interfaces/IJoeRouter01.sol (Base): 2; projects/contracts/interfaces/IJoeRouter02.sol (Base): 2; projects/contracts/interfaces/ILockupHell.sol (Base): 2; projects/contracts/interfaces/IgSGX.sol (Base): 2; projects/contracts/interfaces/Isgx.sol (Base): 2; projects/contracts/GovernanceSGX.sol (Base): 2; projects/contracts/Subgenix.sol (Base): 2; projects/contracts/VaultFactory.sol (Base): 2; projects/contracts/lockupHell.sol (Base): 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.

Alleviation

[CertiK] - The Subgenix team have resolved this issue by locking all of the contracts at version 0.8.4. The commits can been seen at this hash https://github.com/Subgenix-Research/Core/commit/b0942a1f15541c87f070e07bdad896aa7de3146c

GIT-01 | Centralization Risk In VaultFactory.sol

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/contracts/VaultFactory.sol (Base): 205~208, 212~215, 219~223, 228~231, 236~239, 244~247, 252~255, 259~262, 266~269, 274~279, 284~296, 447~475, 587~596, 758~767; VaultFactory.sol (Update1): 207, 278~279	Acknowledged

Description

In the contract `VaultFactory` the role `_owner` has authority over the functions shown in the diagram below.

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and implement two attacks that would lead to the projects demise.

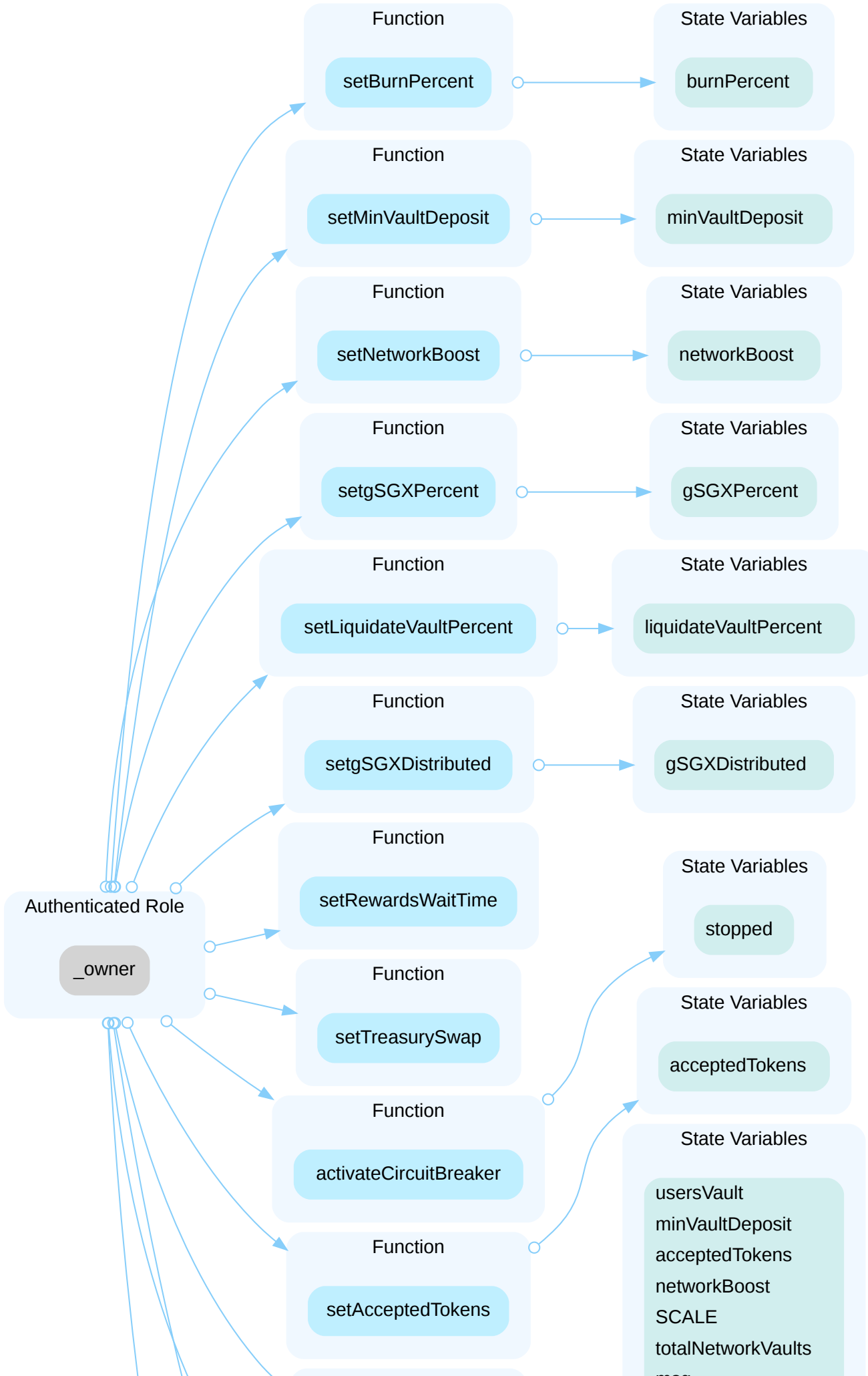
Denote Oscar as the hacker

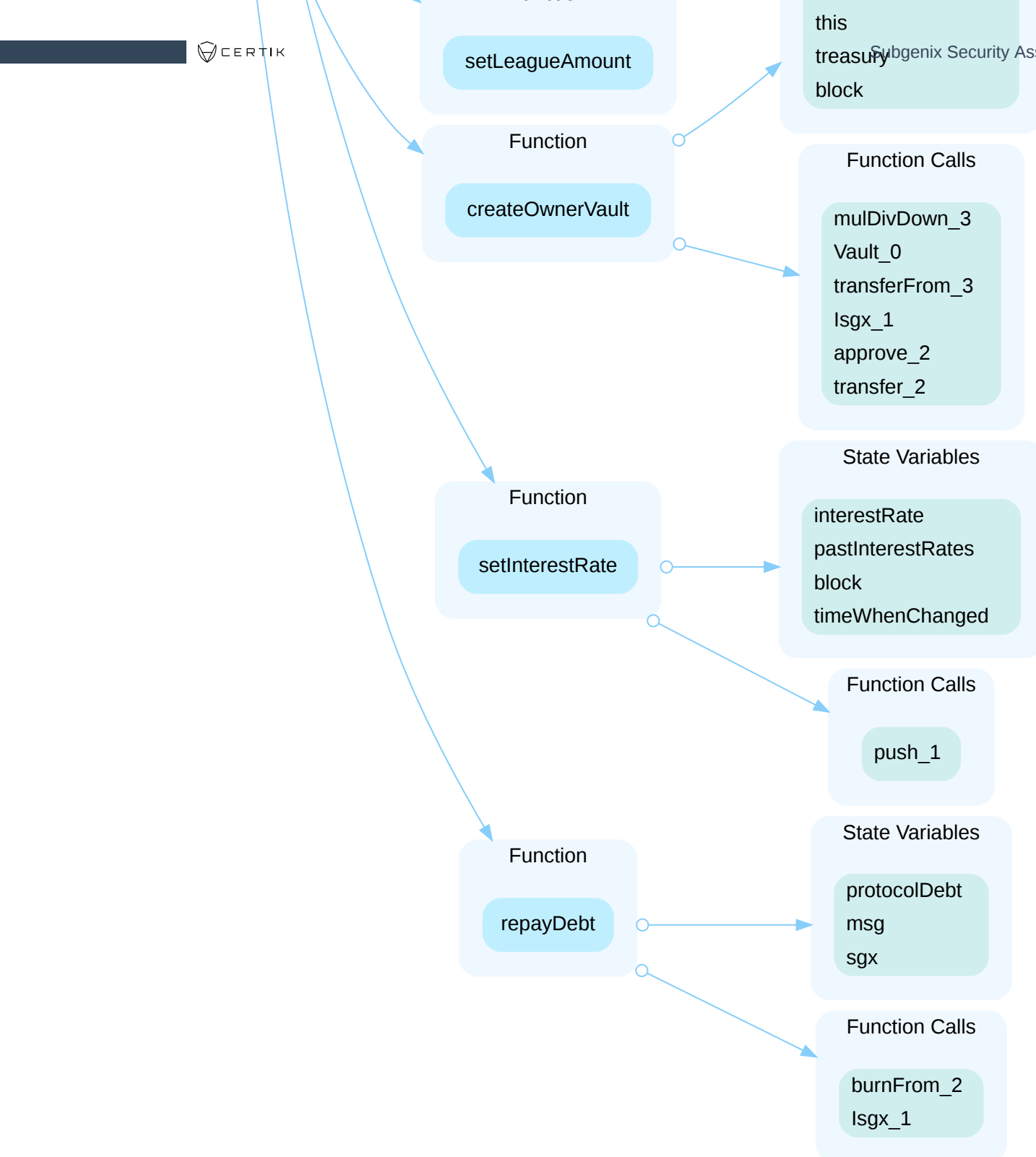
1st Attack

Many `SGX` tokens can be created. Oscar will minimize the time to wait to receive staking rewards by calling `setRewardsWaitTime()`. Then he will call the function `setNetworkBoost` and set the boost to 1000% so that when he stakes his tokens, a larger amount of tokens will be saved in his vault. Finally, Oscar will minimize the variables `burnPercent`, and maximize `liquidateVaultPercent` to maximize his profits.

2nd Attack

Oscar can pause the contract and then renounce ownership via the functions `setCircuitBreaker()` and `renounceOwnership()`. With this attack users cannot deposit liquidity into vaults nor create vaults.


Function
setBurnPercent
State Variables
burnPercent
Function
setMinVaultDeposit
State Variables
minVaultDeposit
Function
setNetworkBoost
State Variables
networkBoost
Function
setgSGXPercent
State Variables
gSGXPercent
Function
setLiquidateVaultPercent
State Variables
liquidateVaultPercent
Function
setgSGXDistributed
State Variables
gSGXDistributed
Function
setRewardsWaitTime
State Variables
stopped
Authenticated Role
_owner
Function
setTreasurySwap
State Variables
acceptedTokens
Function
activateCircuitBreaker
State Variables
usersVault
minVaultDeposit
acceptedTokens
networkBoost
SCALE
totalNetworkVaults
Function
setAcceptedTokens



Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be

improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multisignature wallets.

These attacks can be mitigated if there was a bound on all of the sensitive variables mentioned above. For example, there should be a bound that limits the owner to set the burn percent to 100.

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

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Subgenix Team] - Issue acknowledged. The team will be using a multi-sign gnosis wallet.

SCK-01 | Centralization Risk In Subgenix.sol

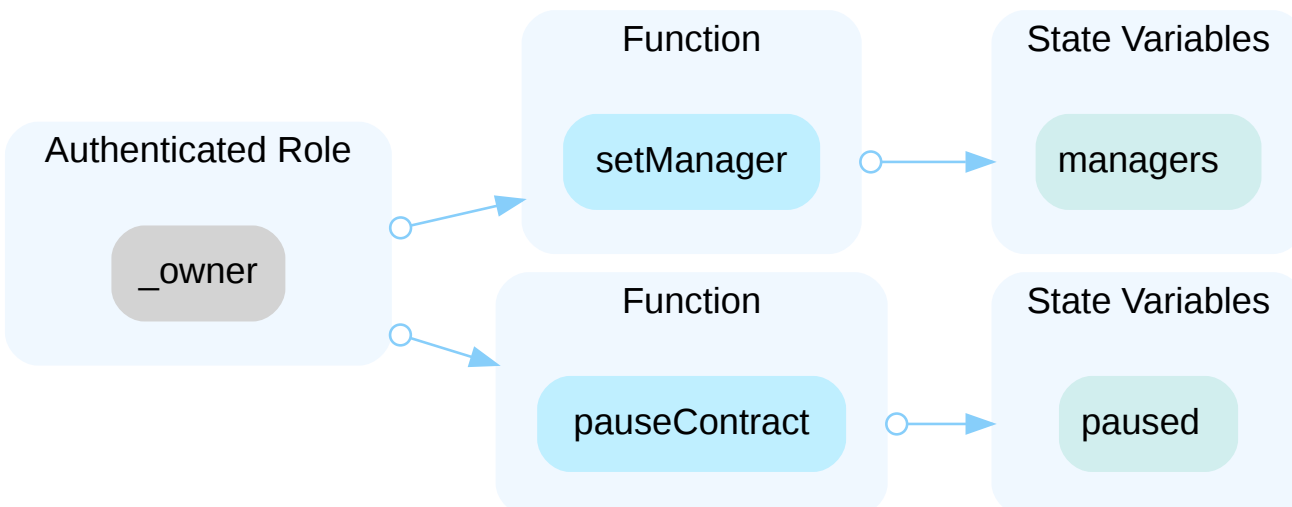
Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/contracts/Subgenix.sol (Base): 94~98, 100~103	Acknowledged

Description

In the contract `Subgenix` the role `_owner` has authority over the functions shown in the diagram below.

Any compromise to the `_owner` account may allow the hacker to take advantage of this authority and set themself as a manager and mint as many tokens as they want.

A compromised `_owner` account may pause the contract indefinitely by first pausing the contract via `pauseContract()` and then renouncing the contract.



Recommendation

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

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

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
  AND
- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles.
  OR
- Remove the risky functionality.

Alleviation

[Subgenix] - Issue acknowledged. We will be using a multi-sign gnosis wallet, and after the liquidity accumulation phase the owner address will renounce the ownership of the contract.

VFC-01 | Missing `require` Statement

Category	Severity	Location	Status
Volatile Code	Minor	projects/contracts/VaultFactory.sol (Base): 220	Resolved

Description

In `ValultFactory.sol` users may stake their `SGX` for future rewards. When a user stakes tokens their vault is updated by adding `(stakedAmount * networkBoost)` to their total balance saved in the vault. If `networkBoost` is zero it will lead to a loss of funds. Therefore we should add a statement in `setNetworkBoost()` that requires `networkBoost > 0`.

Also, `networkBoost` is initialized to zero. The functions `createInVault()`, `createOwnerVault()` and `depositInVault()` handle updating a users balance. These functions should also contain a check that the `networkBoost` is not zero before users partake in staking.

Recommendation

We recommend including a require statement in `setNetworkBoost()` that checks `boost` is greater than zero. The following line is suffice

```
require(boost >= 1e18, "Boost is too small");
```

Alleviation

[CertiK] - The Subgenix team have resolved this issue by following our recommendation above. The commit can been on 235 here https://github.com/Subgenix-Research/Core/commit/b0942a1f15541c87f070e07bdad896aa7de3146c

VFC-02 | Misleading Comments

Category	Severity	Location	Status
Coding Style	Informational	projects/contracts/VaultFactory.sol (Base): 348, 355	Resolved

Description

The comment on line 348 in `VaultFactory.sol` states that `SGX` should be deposited. However, on line 355, there is a possibility of allowing different tokens if `acceptedToken[token] = true`;.

Recommendation

We recommend the team to update the comment and inform us if you only wish to have `SGX` deposited.

Alleviation

[CertiK] - The Subgenix team have replaced the original comment with the following comment, "/// @notice Deposits `amount` of specified token in the vault." This comment reflects the codebase because the function `deposit()` has a check if the token passed is an `acceptedToken`. The changes can be seen at this commit https://github.com/Subgenix-Research/Core/commit/b0942a1f15541c87f070e07bdad896aa7de3146c

VFC-03 | Unclear Use Of `approve()`

Category	Severity	Location	Status
Inconsistency	Informational	projects/contracts/VaultFactory.sol (Base): 340, 342	Resolved

Description

When a user creates a vault a portion of their tokens are transferred to the treasury address. The transfer can be seen on line 342 in `VaultFactory.sol`. The confusion stems from the `approve` statement on line 340. The need for this approve statement is not clear.

Recommendation

Please explain the need of this function call to our team.

Alleviation

[CertiK] - As expected the use of the `approve()` function was not needed. The Subgenix team have removed the `approve` statement. This change does not effect the functionality of the codebase nor introduces and security vulnerabilities. The commits can be seen here https://github.com/Subgenix-Research/Core/commit/b0942a1f15541c87f070e07bdad896aa7de3146c.

VFC-04 | Modifier Does Not Revert Transactions

Category	Severity	Location	Status
Language Specific	Informational	projects/contracts/VaultFactory.sol (Base): 64	Resolved

Description

Modifier VaultFactory.stopInEmergency() (VaultFactory.sol#64) does not revert

File: projects/contracts/VaultFactory.sol (Line 64, Contract `VaultFactory`)

```
    modifier stopInEmergency { if (!stopped) _; }
```

Function calls where `stopInEmergency` is false do not revert.

Recommendation

It seems that this implementation is intended by the team but we want the team to affirm the construction.

Alleviation

[Subgenix Team] - We changed this modifier to a a simple boolean value in the commit hash b0942a1f15541c87f070e07bdad896aa7de3146c.

It was created as a circuit breaker in case anything went wrong in the vaultFactory we would be able to pause all the functions containing this modifier (now just a boolean check).

[CertiK] - In the commit given above, the function `depositInVault()` does not seem to have a modifier that will act as circuit breaker. Please review the commit or give the line where the boolean can be found.

[CertiK - 4/22] A boolean value `stopped` has been added to the project to act as a circuit breaker. This clears up our concerns of a missing circuit breaker.

Remark It is possible that a compromised owner can paused the contract indefinitely by activating the circuit breaker and renouncing ownership of the contract. Therefore we highly advise the team to implement a multi-sig wallet.

VFC-05 | Unused Return Value

Category	Severity	Location	Status
Volatile Code	Informational	projects/contracts/VaultFactory.sol (Base): 550, 551	Resolved

Description

The return value of an external call is not stored in a local or state variable.

File: projects/contracts/VaultFactory.sol (Line 550, Function `VaultFactory.swapSGXforAVAX`)

```
        joeRouter.swapExactTokensForAVAX(toTreasury, 0, path, treasury, block.timestamp);
```

File: projects/contracts/VaultFactory.sol (Line 551, Function `VaultFactory.swapSGXforAVAX`)

```
        joeRouter.swapExactTokensForAVAX(toResearch, 0, path, research, block.timestamp);
```

Recommendation

We recommend checking or using the return values of all external function calls.

Alleviation

[CertiK] - There is an inclusion of an array `amounts` and we check if the element `amounts[0]` is greater than zero. It seems that `amounts[0]` should be the input token amount which should always be greater than zero. Therefore the function would always revert. Can you please explain this implementation?

[Subgenix - 4/21/22] This was a mistake made by the team, we changed the if function to a require. Commit hash of the change: 6e73dffe2c1c3584eb935ec2c3531a775428a245

[CertiK] - The change to a `require` statement does resolve the issue.

Clarification of Fix Before the swap is implemented we should ensure that the input token amount is greater than zero. Originally, the code would check if the input amount is greater than zero and would revert. However, that is not the intended implementation. Instead the team correctly added a require statement to ensure the input amount is positive. The changes can be seen here https://github.com/Subgenix-Research/Core/commit/6e73dffe2c1c3584eb935ec2c3531a775428a245

VFC-06 | Function Should Be Declared External

Category	Severity	Location	Status
Gas Optimization	Informational	projects/contracts/VaultFactory.sol (Base): 604	Resolved

Description

The functions which are never called internally within the contract should have external visibility for gas optimization.

File: projects/contracts/VaultFactory.sol (Line 604, Contract `VaultFactory`)

```
    function claimRewards(address user) public nonReentrant {
```

Recommendation

We advise to change the visibility of the aforementioned functions to `external`.

Alleviation

[CertiK] - The Subgenix team have resolved the issue by following the recommendation above. The commits can be seen on line 619 here https://github.com/Subgenix-Research/Core/commit/b0942a1f15541c87f070e07bdad896aa7de3146c

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.

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.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

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.